

Introdução a String

Mauricio Ferste

Strings

Formatação de Strings

Operações, Funções e

Métodos Exercícios

- Uma String é uma sequência de caracteres.
- Em Python, Strings são representadas como listas imutáveis de caracteres.
- Podemos representar uma String como sequências 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
3 syntax 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 print("\
6 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:
12 string index
13 out of range
```

- 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

```
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
```

1
1

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#format-spec>
- Iremos focar na formatação valores do tipo inteiro (`int`) e real (`float`).

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
```

Formatação de Strings

- 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 número 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 número reais:

```
1 pi = "O valor de pi é: {0:.4f}"
2
3 print(pi.format(3.14159265359))
4     # O valor de pi é: 3.1416
5 notas = "A média das notas da
6 turma foi {0:.2f}."
7
8 print(notas.format(8.7525))
9     # A média das notas da turma
10    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,
6 28.765))    # 07/05/2020: 28.8C
```

Operações, Funções e Métodos

Concatenação de Strings

- O operador + concatena duas Strings.

```
1 msg = "hello"
2 msg2 = "y" + msg[1:5] + "w"
3 print (msg2)
4     # yellow
```

- O operador * faz concatenações múltiplas da mesma String.

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

```
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

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

Comparação de Strings

- 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

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

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 str = "aeiou"
2 lista = list(str)
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 trocas 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

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

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

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

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

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

Exemplo

- 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
6 vogal: o    # A String possui
7 a vogal: i    # A String
8 possui a vogal: o
```


Hackaton

Organização em equipes!

Competição saudável

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.