Algoritmos e Programação de Computadores

Strings

Prof. Edson Borin
Instituto de Computação (IC/Unicamp)

Agenda

- Criação de strings e acesso aos caracteres da string com índices
- Quebra de linha (\n) e outros caracteres especiais
- Operadores + e *
- Percorrer caracteres da strings como listas
- Operador de slicing
- Método strip
- Operador in
- Comparação de strings
- Comparação de strings
- Método isnumeric
- Métodos index e find
- Método count
- Método split
- Método replace
- Função list
- Método join
- Métodos lower e upper
- Função len
- Método format

Strings: criação e acesso aos caracteres

- Strings em Python são listas imutáveis de caracteres.
- Strings são representadas por sequências de caracteres entre aspas simples `'` ou entre aspas duplas `"`.

```
a = "20 de Abril tem prova."
a
'20 de Abril tem prova.'
b = 'Fizeram a atividade conceitual?'
b
'Fizeram a atividade conceitual?'
c = "Que vida \"fácil\""
c
'Que vida "fácil"'
```

Strings: criação e acesso aos caracteres

- Strings em Python são listas imutáveis, portanto pode-se acessar posições de uma string de forma usual.

```
a = "20 de Abril tem prova."
a[0]
'2'
a[0] = "1"
```

```
---------------------------------------------------------------
TypeError                 Traceback (most recent call last)
<ipython-input-13-9ab1dda42293> in <module>()
----> 1 a[0] = "1"

TypeError: 'str' object does not support item assignment
```

Caracteres especiais: \n, \t, ...

- O caractere `'\n'` pode fazer parte de uma string e ele só causa a mudança de linha no comando **`print`**.

```
a = 'Fizeram\na\natividade\nconceitual?'
a
'Fizeram\na\natividade\nconceitual?'
```

```
a = 'Fizeram\na\natividade\nconceitual?'
print(a)
Fizeram
a
atividade
conceitual?
```

Caracteres especiais: \n, \t, ...

- O caractere `'\t'` pode fazer parte de uma string e ele representa o caractere de tabulação (tab).

```
print('A:\t12')
print('Aa:\t11')
print('AaA:\t10')
```

```
A:  12
Aa: 11
AaA:    10
```

Caracteres especiais: \n, \t, ...

- Os caracteres `'` e `"` podem ser adicionados em strings com o auxílio do caractere de escape (`'\'`)

```
print('Marca d\'água')
print('\'Entre aspas simples\'')
print('\"Entre aspas duplas\"')
```

```
Marca d'água
'Entre aspas simples'
"Entre aspas duplas"
```

Operadores + e *

- O operador + concatena 2 strings, e o operador * repete a concatenação (como em listas).

```
a = "20 de Abril tem prova."
b = 'Fizeram a atividade conceitual?'
a + b
'20 de Abril tem prova.Fizeram a atividade conceitual?'
```

```
b = 'Fizeram a atividade conceitual?\n'
print(3*b)
Fizeram a atividade conceitual?
Fizeram a atividade conceitual?
Fizeram a atividade conceitual?
```

Percorrer caracteres da strings como listas

- Strings podem ser processadas como listas, podendo por exemplo ter seus elementos percorridos num laço **for**.
- Exemplo: Ler uma string e imprimir a inversa.

```
string = input("Digite um texto: ")
inversa = ""
for x in string:
    inversa = x + inversa
print(inversa)
```

Operador de *slicing*

- O operador **slice** (fatiar) pode ser usado para formar uma nova string a partir de um subconjunto de caracteres da string original.
- Pode-se fatiar (slice) strings usando `[inicio:fim-1:passo]`.

```
a = "20 de Abril tem prova."
a[6:11]
'Abril'
a[6:11:2]
'Arl'
a[::-1]
'.avorp met lirbA ed 02'
```

- A string vazia é representada como `''` ou `""`.

Método strip

- O método `strip` retorna uma string sem os brancos e mudança de linhas **no início e no final** de uma string.

```
b = "\n  Fizeram a atividade conceitual?  \n\n"
print("["+b+"]")
c = b.strip()
print("["+c+"]")
```

```
[
  Fizeram a atividade conceitual?

]
[Fizeram a atividade conceitual?]
```

Operador in

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

```
"atividade" in "Fizeram a atividade conceitual?"
True

"idade" in "Fizeram a atividade conceitual?"
True

"Abril" in "Fizeram a atividade conceitual?"
False
```

Comparação de strings

- Os operadores de comparação == e != podem ser usados para comparar strings

```
a = "Python"
b = "Py" + "thon"
c = "p" + "ython"
print( a == b )
print( a == c )
print( b != c )
```

Comparação de strings

- Os operadores booleanos == e != podem ser usados para comparar strings

```
a = "Python"
b = "Py" + "thon"
c = "p" + "ython"
print( a == b )
print( a == c )
print( b != c )
```

```
True
False
True
```

Método isnumeric()

- O método isnumeric() testa se todos os caracteres são dígitos numéricos.

```
A = "1234"
print(A.isnumeric())
B = "test123"
print(B.isnumeric())
```

```
True
False
```

Métodos index e find

- O método `index` retorna onde a substring começa na string.

```
a = "Fizeram a atividade conceitual?"
a.index("atividade")
10

a.index("abril")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

- O método `index` causa um erro quando a substring não ocorre na string.

Métodos index e find

- O método `find` retorna onde a substring começa na string.

```
a = "Fizeram a atividade conceitual?"
a.find("atividade")
10

a.find("abril")
-1
```

- O método `find` retorna `-1` quando a substring não ocorre na string.

Método count

- O método `count` conta o número de ocorrências de uma substring em uma string.

```
a = "oitocentos e oitenta e oito"
a.count("oi")
3

a.count("e")
4
```

Método split

- O método `split(sep)` separa uma string usando **sep** como separador. Retorna uma lista das substrings.

```
numeros = "1;; 2 ;  3"
print(numeros.split(";"))
```

```
['1', '', ' 2 ', '  3']
```

- Podem haver substrings vazias no retorno de `split(sep)`.

Método split

- O método `split()` considera um ou mais espaços consecutivos como um único delimitador (\t e \n também são considerados espaços)

```
msg = "  \n \t \t Fizeram \n  a \t atividade   conceitual?"
print("msg = ["+msg+"]")
print("msg.split() = ",msg.split())
```

```
msg = [
          Fizeram
  a       atividade   conceitual?]
msg.split() =  ['Fizeram', 'a', 'atividade', 'conceitual?']
```

Algoritmos e Programação de Computadores

Strings (Continuação)

Prof. Edson Borin
Instituto de Computação (IC/Unicamp)

Método replace

- O método `replace` serve para trocar **todas** as ocorrências de uma substring por outra em uma string.

```
a = "Fizeram a atividade conceitual?"
b = a.replace("conceitual","teórica")
print("b = ["+b+"]")
c = a.replace("conceitual","")
print("c = ["+c+"]")
```

```
b = [Fizeram a atividade teórica?]
c = [Fizeram a atividade ?]
```

Função list

- Podemos usar a função `list` para transformar uma string em uma lista onde os itens da lista correspondem aos caracteres da string.

```
numeros = "1; 2 ;  3"
list(numeros)
['1', ';', ' ', '2', ' ', ';', ' ', ' ', '3']

list("atividade")
['a', 't', 'i', 'v', 'i', 'd', 'a', 'd', 'e']
```

Método join

- O método `join` recebe como parâmetro uma sequência ou lista, e retorna uma string com a concatenação dos elementos da sequência/lista.

```
l = list("atividade")
l
['a', 't', 'i', 'v', 'i', 'd', 'a', 'd', 'e']

"".join(l)
'atividade'

"-".join(l)
'a-t-i-v-i-d-a-d-e'
```

Métodos lower e upper

- O método `lower` (`upper`) converte todos os caracteres da string para caixa baixa (alta).

```
str = "Atividade"
str1 = str.upper()
str2 = str.lower()
print(str)
Atividade
print(str1)
ATIVIDADE
print(str2)
atividade
```

Função len

- O método `len()` conta o número de caracteres na string.

```
str = "MC102"
print(str,"tem",len(str),"caracteres.")
```

```
MC102 tem 5 caracteres.
```

Função len

- O método `len()` conta o número de caracteres na string.

```
str = "\tMC102\n"
print(str,"tem",len(str),"caracteres.")
```

```
   MC102
 tem 7 caracteres.
```

\t e \n contam como caracteres.

Método format

- A função `format`() pode ser usada para formatar uma string.

```
str = "Frutas: {0}, {1}, {2}"
print(str.format("maçã","mamão","banana"))
```

```
Frutas: maçã, mamão, banana
```

- Epecificação completa:
  https://docs.python.org/3/library/string.html#formatspec

Strings: Resumo

Método	Parâmetros	Descrição
`strip`	nenhum	Retorna uma string removendo caracteres em branco do início e do fim. Ex: `a.strip()`
`isnumeric`	nenhum	Retorna True se a string contém apenas dígitos numéricos, do contrário False.
`index`	substring	Retorna o índice onde a substring começa na string. Causa erro se não encontrar.
`find`	substring	Retorna o índice onde a substring começa na string. Retorna -1 se não encontrar. Ex: `a.find("texto")`
`count`	substring	Retorna o número de ocorrências de uma substring. Ex: `a.count("as")`
`split`	nenhum	Separa uma string usando sep como separador e retorna uma lista das substrings. Ex: `a.split()`
`replace`	substring1, substring2	Substitui todas as ocorrências de uma substring por outra. Ex: `a.replace("prova","teste")`
`join`	substring	Retorna uma string com a concatenação dos elementos da sequência/lista. Ex: `"".join(a)`
`Upper / lower`	nenhum	Retorna uma string toda em maiúsculas/minúscula. Ex: `a.upper() a.lower()`

Strings: Resumo

Operador / função	Descrição
`+`	Cria uma string nova a partir da concatenação de duas strings distintas. Ex: `"oi" + " mundo"`
`*`	Cria uma string nova a partir de múltiplas concatenações de uma string. Ex: `" oi" * 3`
`slicing`	Cria uma string nova a partir de um subconjunto de caracteres de uma string original.
`in`	Verifica se uma substring é parte de uma string. Ex: `"to" in "texto"`
`==, !=, <, <=, ...`	Operadores relacionais podem ser usados para comparar strings. Lembrar que todas as letras maiúsculas precedem as letras minúsculas. Ex: `"Azul" < "azul"`
`list`	Transforma uma string em uma lista onde os itens da lista correspondem aos caracteres da string. Ex: `list("texto")` ou `list(a)`
`len`	Retorna o tamanho (números de caracteres) de uma string: `len("texto")`

Exercícios

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.
  - Primeiramente removemos do texto todos os sinais de pontuação.

```
texto = input("Digite um texto: ")
pontuacao = [".", ",", ":", ";", "!", "?"]

# remove os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")
```

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.
  - Depois usamos a função `split` para separar as palavras.

```
texto = input("Digite um texto: ")
pontuacao = [".", ",", ":", ";", "!", "?"]

# remove os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")

# split devolve lista com palavras como itens
numero_palavras = len(texto.split())
print("Número de palavras:", numero_palavras)
```

Exercício: Palíndromo

- Faça um programa que lê uma string e imprime “Palíndromo” caso a string seja um palíndromo e “Não é palíndromo” caso não seja.
    - Assuma que a entrada não tem acentos e que todas as letras são minúsculas.

- Obs: Um *palíndromo* é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (espaços em brancos são descartados).
    - Exemplos de palíndromo: “ovo”, “reviver”, “mega bobagem”, “anotaram a data da maratona”

Exercício: Palíndromo

- Faça um programa que lê uma string e imprime “Palíndromo” caso a string seja um palíndromo e “Não é palíndromo” caso não seja.

Entrada	Saída
ovo	Palíndromo

Entrada	Saída
prova	Não é palíndromo

Entrada	Saída
anotaram a data da maratona	Palíndromo

Exercício: Palíndromo

```
texto = input("Digite um texto: ")

# remove os espaços em branco
texto = texto.replace(" ","")

# inverte a string
texto_inverso = texto[::-1]

# verifica se texto é igual ao texto_inverso
if (texto == texto_inverso):
    print("Palíndromo")
else:
    print("Não é palíndromo")
```

Exercício: Palíndromo

- Faça uma nova versão que aceita como palíndromo mesmo que as letras correspondentes sejam maiúsculas e minúsculas.
  - Exemplo: “Ovo”, “Anotaram a Data da Maratona” devem ser também palíndromo.

Exercício: Palíndromo

```
texto = input("Digite um texto: ")

# remove os espaços em branco
texto = texto.replace(" ","")

# inverte a string
texto_inverso = texto[::-1]

# verifica se texto é igual ao texto_inverso
if (texto.lower() == texto_inverso.lower()):
    print("Palíndromo")
else:
    print("Não é palíndromo")
```

Exercício: Palíndromo

```
texto = input("Digite um texto: ")

# remove os espaços em branco
texto = texto.replace(" ","")

# inverte a string
texto_inverso = texto[::-1]

# verifica se texto é igual ao texto_inverso
if (texto.lower() == texto_inverso.lower()):
    print("Palíndromo")
else:
    print("Não é palíndromo")
```

Exercício: Data por extenso

- Faça um programa que solicite a data de nascimento (dd/mm/aaaa) do usuário e imprima a data com o nome do mês por extenso.

Entrada	Saída
`16/12/1982`	`16 de dezembro de 1982`

Exercício: Data por extenso

```
data = input("Digite a data de nascimento (dd/mm/aaaa): ")
```

Exercício: Data por extenso

```
data = input("Digite a data de nascimento (dd/mm/aaaa): ")

# separa a data pelo caractere "/"
lista_data = data.split("/")

# transforma o número do mês em mês por extenso
meses = ["janeiro", "fevereiro", "março", "abril", "maio",
         "junho", "julho", "agosto", "setembro", "outubro",
         "novembro", "dezembro"]

mes_extenso = meses[int(lista_data[1])-1]
print(lista_data[0] + " de " + mes_extenso + " de " + lista_data[2])
```

Exercício: Conta espaços e vogais

- Faça um programa que conta espaços e vogais. Dado um texto (sem acento) informado pelo usuário, conte:
  - Quantos espaços em branco existem no texto.
  - Quantas vezes aparecem as vogais a, e, i, o, u.

Entrada	Saída
`18 de Abril tem revisao para a prova`	`espaços: 7` `a: 6 e: 3 i: 2 o: 2 u: 0`

Solução 1

Exercício: Conta espaços e vogais

```
texto = input("Digite um texto: ")

# conta espaços em branco
numero_espacos = texto.count(" ")
print("espaços:", numero_espacos)

# conta vogais
vogal_a = texto.lower().count("a")
vogal_e = texto.lower().count("e")
vogal_i = texto.lower().count("i")
vogal_o = texto.lower().count("o")
vogal_u = texto.lower().count("u")
print("a:",vogal_a,"e:",vogal_e,"i:",vogal_i,"o:",vogal_o,"u":
      ,vogal_u)
```

Solução 2

Exercício: Conta espaços e vogais

```
texto = input("Digite um texto: ")

# conta espaços em branco
numero_espacos = texto.count(" ")
print("espaços:", numero_espacos)

# conta vogais
vogais = ["a", "e", "i", "o", "u"]
for v in vogais:
    numero_vogais = texto.lower().count(v)
    print(v,":",numero_vogais, end=" ")
```

Solução 3

Exercício: Conta espaços e vogais

```
texto = input("Digite um texto: ")

# conta espaços em branco
numero_espacos = texto.count(" ")
print("espaços:", texto.count(" "))

# conta vogais
vogais = ["a", "e", "i", "o", "u"]
for v in vogais:
    numero_vogais = texto.lower().count(v)
    print(v,":",texto.lower().count(v), end=" ")
```

Solução 3

Exercício: Conta espaços e vogais

```
texto = input("Digite um texto: ")

# conta espaços em branco
print("espaços:", texto.count(" "))

# conta vogais
vogais = ["a", "e", "i", "o", "u"]
for v in vogais:
    print(v,":",texto.lower().count(v), end=" ")
```

Exercício: Jogo da Forca

- Faça um jogo da forca. O programa lerá uma lista de palavras e escolherá uma aleatoriamente. O jogador poderá errar 6 vezes antes de ser perder.

```
Digite uma letra: a
-> Você errou pela 1a vez. Tente de novo!

Digite uma letra: o
A palavra é: _ _ _ _ o

Digite uma letra: e
A palavra é: _ e _ _ o

Digite uma letra: s
-> Você errou pela 2a vez. Tente de novo!
```

```
import random # importa o módulo random
palavras = input("Digite as palavras: ")
palavras = palavras.split(" ")

# pega um número aleatoriamente entre 0 e número de palavras
palavra_sorteada = palavras[random.randrange(0,len(palavras))]
```

Complete o programa ...

Referências & Exercícios

- https://wiki.python.org.br/ExerciciosComStrings: 14 exercícios =)
- https://wiki.python.org.br/ExerciciosListas: 24 exercícios =)

- https://panda.ime.usp.br/pensepy/static/pensepy/08-Strings/strings.html
- https://panda.ime.usp.br/pensepy/static/pensepy/09-Listas/listas.html

Créditos

Os *slides* deste curso foram baseados nos slides produzidos e cedidos gentilmente pela Professora Sandra Ávila, do Instituto de Computação da Unicamp.