Algoritmos e Programação de Computadores

Strings

Ref.: material original (1o S., T. KLMN). por **Profa. Sandra Avila**, Instituto de Computação (IC/ Unicamp)

Agenda

- Revisão ex. Aula 10, listas
- Strings
  - Operações
  - Funções
  - Métodos
- Exercícios

Revisão Exercício Listas
Aula 10

Exercício

- Faça um programa que:
  - Lê dois vetores com 5 inteiros cada.
  - Checa quais elementos do segundo vetor são iguais a algum elemento do primeiro vetor.
  - Se não houver elementos em comum, o programa deve informar isso.

Entrada	Saída
`[1, 2, 3, 4, 5]` `[0, 7, 6, 10, 3]`	`3`

Entrada	Saída
`[1, 2, 3, 4, 5]` `[0, 7, 6, 10, 8]`	`Não tem.`

```
x = []
y = []
for i in range(5):
    x.append(int(input()))
print()

for i in range(5):
    y.append(int(input()))
print()

um_elemento_comum  = False

#Assumimos que não temos elementos comuns
for i in range(len(x)):
    for j in range(len(y)):
        if (x[i] == y[j]):
            um_elemento_comum  = True # há elemento comum
            print(str(x[i]))

if not um_elemento_comum:
    print("Não tem elemento comum.")
```

```
x = []
y = []
for i in range(5):
    x.append(int(input()))
print()

for i in range(5):
    y.append(int(input()))
print()

um_elemento_comum  = False

#Assumimos que não temos elementos comuns
for a in x:
    for b in y:
        if (a == b):
            um_elemento_comum  = True # há elemento comum
            print(str(a))

if not um_elemento_comum:
    print("Não tem elemento comum.")
```

Strings

Strings

- *String* (literalmente) significa sequência, série, cadeia (de caracteres).
- String é um tipo particular de dados, suportado por Python, ***str***, porém, o tipo *caractere* não é suportado por Python.

A fim de representar texto, em versões anteriores a Python3, caracteres foram codificados como uma sequência de "Bytes" individuais, utilizando o conjunto **ASCII** (American Standard Code for Information Interchange), onde cada caractere é formado por 7-bits, limitando a representação a 128 símbolos. A partir da versão 3, utiliza-se a codificação **UNICODE**, que permite representar texto universal(i.e.,texto em qualquer escrita atual > 137K caracteres, ex. ideogramas/emoji).

Strings

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

```
a = “07 de Setembro eh feriado."
ou
‘07 de Setembro eh feriado.'
b = “Estarao a viajar?”
ou
‘Estarao a viajar?'
c = "Que vida \"fácil\""
ou
'Que vida "fácil"'
```

Strings

- Strings em Python são sequências imutáveis, podem-se acessar posições de uma string usando índices (um caractere é um *substring*).

```
a = “07 de Setembro eh feriado."
a[1]
'7'
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
```

Strings

- O caractere '\n' <NL>(New-Line) pode fazer parte de uma string e causa <CR>+<LF>, <CR> é posicionamento do cursor na 1a. Coluna, e LF(line-feed) é a mudança de linha.
  <CR> é executado pelo caractere '\r', e apenas retorna o cursor à 1a coluna.

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

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

Strings: Operações, Funções e Métodos

- O operador + concatena 2 strings, e o operador * repete a concatenação (de forma semelhante à utilização em listas).

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

Strings via Loops

- Strings podem ser processadas via *loops* (laços), de forma semelhante a uma lista, sendo possível percorrer os seus elementos por meio de um *loop*/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)
```

Strings: Operações, Funções e Métodos

- A função **slice** (fatiar) devolve a string entre duas posições dadas.
- Podem-se fatiar separar substrings, via `[inicio:fim-1:passo]`.

```
a = "07 de Setembro tem feriado."
a[6:14]
'Setembro'
a[6:14:2]
'Stmr'
a[::-1]
'.odairef met orbmeteS ed 70'
```

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

Strings: Operações, Funções e Métodos

- O método `strip` retorna uma string sem os brancos e caracteres `\n` (`NL-newline`, mudança de linhas) **no início e no final** de uma

```
b = "Fizeram a atividade conceitual?"
b
'\n Fizeram a atividade conceitual? \n'

b.strip()
'Fizeram a atividade conceitual?'
```

Strings: Operações, Funções e Métodos

- 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
```

Strings: Operações, Funções e Métodos

- 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.

Strings: Operações, Funções e Métodos

- O método `split(sep)` separa uma string usando o caractere **sep** como separador. O método retorna uma lista contendo as substrings.

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

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

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

Strings: Operações, Funções e Métodos

- 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']
```

Strings: Operações, Funções e Métodos

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

```
a = "Fizeram a atividade conceitual?"
a.replace("conceitual","teórica")
'Fizeram a atividade teórica?'

a = "Fizeram a atividade conceitual?"
a.replace("conceitual","")
'Fizeram a atividade ?'
```

Strings: Operações, Funções e Métodos

- O método join recebe como parâmetro uma sequência ou lista, e retorna uma *string* com a concatenação dos elementos da sequência/lista. A sintaxe, para uma lista com identificador de lista id_lista e caractere de separação <separa> é : <separa>.join(id_lista)

```
lista1 = list("atividade")
lista1
['a', 't', 'i', 'v', 'i', 'd', 'a', 'd', 'e']
# ex. string gerado usando o caractere vazio “” como separador
"".join(lista1)
'atividade'
```

Exercícios

Exemplo: Contador de Palavras

- Escrever um programa na linguagem Python que conte o número de palavras em um texto.
- Como entrada, um texto será digitado de forma interativa no teclado, incluindo alguns carateres de pontuação, tal como os definidos na lista definida a seguir:
  - `Lista com caracteres de pontuação: [".", ",", ":", ";", "!", "?"]`

Exemplo: Contador de Palavras

- Programa Python que conta o número de palavras em um texto.
  - Inicialmente é ingressado texto incluindo diversos sinais de pontuação.
  - Na sequencia, removem-se do texto todos os sinais de pontuação.

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

# removendo os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")
print("Número de palavras:", numero_palavras)
```

Exemplo: Contador de Palavras

- programa Python que conta o número de palavras em um texto.
  - Utilizando o método split para separar as palavras, gerando uma lista
  - Aplica-se a função len à lista gerada, para determinar o seu comprimento

```
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

- Escreva um programa que leia uma string e imprima “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”

Referências & Exercícios

- Os slides dessa aula foram baseados no material de MC102 dos Prof. Sandra Ávila e Eduardo Xavier (IC/Unicamp)
- https://wiki.python.org.br/ExerciciosComStrings: 14 exercícios =)
- https://panda.ime.usp.br/pensepy/static/pensepy/08-Strings/strings.html