MC-102 — Aula 14
Strings

Prof. Luiz F. Bittencourt

Turmas QR

Instituto de Computação – Unicamp

2019

Conteúdo adaptado de slides fornecidos pelo Prof. Eduardo Xavier.

- Inscrições: bit.ly/meeting-jam
- Mais informações no evento no Facebook: https://www.facebook.com/events/104708279215267/
- Próximo final de semana no IC.

Roteiro

1. Strings
   - Strings; operações, funções e métodos
2. Processamento de Texto
3. Exercícios

Strings

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

```
>>> a = "Qwerty de Asdf"
>>> a
'Qwerty de Asdf'
>>> a = 'Qwerty de Asdf'
>>> a
'Qwerty de Asdf'
>>> c = 'Joe\'s Garage'
>>> c
"Joe's Garage"
```

Strings

- Strings são listas imutáveis, portanto pode-se acessar posições de uma string da forma usual:

```
>>> a='querty'
>>> a
'querty'
>>> a[2]
'e'
>>> a[-1]
'y'
```

Strings

- Strings são imutáveis:

```
>>> a='qwerty'
>>> a[2]
'e'
>>> a[2]='z'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
```

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

```
>>> aa='abc\ndef'
>>> aa
'abc\ndef'
>>> print(aa)
abc
def
```

Strings: operações, funções e métodos

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

```
>>> 'qwerty'+'poiuy'
'qwertypoiuy'
>>> 3*'abc'
'abcabcabc'
```

- A função **slice** devolve a string entre duas posições dadas.

```
>>> a='qwerty'
>>> a[2:4]
'er'
```

- A string vazia é representada como ' '.

Strings como listas (imutáveis)

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

```
st = input("Digite um texto:")
inv = ''
for x in st:
  inv = x + inv
print(inv)
```

- Note que cada caractere, em ordem, é adicionado no início de **inv** de tal forma que o último caractere de **st** será o primeiro de **inv**.

Strings: operações, funções e métodos

- A função **input** (já vista) lê e retorna uma string.
- O método **strip** retorna uma string sem os brancos e mudança de linhas no início e final de uma string.

```
>>> aa=' \n  abcndef   \n'
>>> aa
' \n  abcndef   \n'
>>> print(aa)

  abcndef

>>> aa.strip()
'abcndef'
```

Strings: operações, funções e métodos

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

```
>>> 'tho' in 'python'
True
>>> 'thor' in 'python'
False
```

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

```
>>> p='python'
>>> p.find('tho')
2
>>> p.find('thor')
-1
```

- O método **find** retorna -1 quando a substring não ocorre na string (veja o exemplo acima).

Strings: operações, funções e métodos

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

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

- O método **split()** separa usando espaço '\n' e tab como **sep**.

```
>>> b="ouviram   do  ipiranga margens"
>>> b.split()
['ouviram', 'do', 'ipiranga', 'margens']
```

- Note que pode haver substrings vazias no retorno de **split()**.

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

Strings: operações, funções e métodos

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

```
>>> a="abcabcdfgabc abc a b c"
>>> a.replace("abc","")
'dfg  a b c'
```

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

```
>>> a="abc\n;abc"
>>> list(a)
['a', 'b', 'c', '\n', ';', 'a', 'b', 'c']
```

- O método **join** recebe como parâmetro uma sequência ou lista, e retorna uma string com a concatenação dos elementos da sequencia/lista utilizando o separador "str"

```
>>> a="abc\n;abc"
>>> l = list(a)
>>> l
['a', 'b', 'c', '\n', ';', 'a', 'b', 'c']
str=""
>>> str.join(l)
'abc\n;abc'
```

Processamento de Texto

- Como exemplo de funções com strings vamos implementar duas funcionalidades básicas de processadores de texto:
  1. Contar o número de palavras em um texto.
  2. Fazer a busca de todas as ocorrências de uma palavra em um texto.

Processamento de Texto

- Primeiramente removemos do texto todos os sinais de pontuação.
- Depois usamos a função split para separar as palavras.

```
st = input("Digite um texto:")
pontuacao = [".", ",", ":", ";", "!", "?"]
for pont in pontuacao: #remove os sinais de pontuação
  st = st.replace(pont," ")

numPal = len(st.split()) #split devolve lista com palavras como itens
print("Num. palavras:", numPal)
```

Processamento de Texto

Achar palavras em um texto.

- Fazer um programa que acha todas as posições de ocorrência de uma palavra (substring) em um texto (uma string).

Exemplo:

```
Texto=a tete tetete
Palavra=tete

A resposta é [2, 7, 9]
```

Processamento de Texto

Ideia do algoritmo:

- Achamos a posição de ocorrência de **subst** em **st** e armazenamos esta na lista **pos**.
- Depois removemos toda parte inicial de **st** até o primeiro caractere onde encontramos **subst**:
  Exe: subst="abc" e st="dfg abcabc", vamos remover "dfg a" ficando st="bcabc".
- Como removemos uma parte inicial de **st** precisamos guardar o seu tamanho em **tamRemovido** pois próximas ocorrências estão deslocadas por este valor na string original.

```
>>> subst="abc"
>>> st="dfg abcabc"
>>> st.find(subst)
4                              #posição da 1o ocorrência
>>> st = st[4+1:]
>>> st
'bcabc'                       #removeu até a 1o ocorrência
>>> tamRemovido=4+1  #tamanho da parte removida
>>> st.find(subst)   #posição da 2o ocorrência
2
>>> pos = [4, tamRemovido+2] #posição da 2o ocorrência considerando o que foi removido
>>> pos
[4, 7]
```

Processamento de Texto

- Lembre-se que o **slicing st[pos:]** devolve uma string contendo todos os caracteres a partir da posição **pos**.
- Usamos isso para remover o início do texto até a primeira ocorrência da palavra.

```
st = input("Entre com um texto:")
subst = input("Entre com uma palavra:")
pos = []
tamRemovido = 0
while subst in st: #Enquanto houver uma ocorrência da palavra em st
  aux = st.find(subst) #acha posicao da 1o ocorrência
  pos.append(aux+tamRemovido) #inclui posição corrigida da palavra
  st = st[(aux+1):] #remove tudo até 1a letra da 1o ocorrência de subst
  tamRemovido = tamRemovido + aux + 1

print(pos)
```

Exercício

- Escreva um programa que lê uma string, e imprime "Palindromo" caso a string seja um palindromo e "Nao Palindromo" caso contrário.
- OBS: Um palindromo 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). Assuma que a entrada não tem acentos e que todas as letras são minúsculas
- Exemplo de palindromo: "saudavel leva duas"
- Faça uma nova versão que aceita como palindromo mesmo que as letras correspondentes sejam maiúsculas e minúsculas. Assim "Saudavel Leva DUas" deve ser também um palindromo.

Exercício

- O usuário entra cinco números separados por brancos. Imprima a média deles.
- O usuário entra com vários números separados por branco ou virgula, por exemplo “3,4 5 6, 9” . Imprima a média deles.