

LÓGICA E LINGUAGEM DE PROGRAMAÇÃO

Curso Técnico Subsequente em Informática
Lucas Sampaio Leite



Strings

- Em Python, uma string é uma sequência de caracteres que pode conter letras, números, símbolos e espaços.
- As strings são utilizadas para representar textos e podem ser definidas com aspas simples ('), aspas duplas ("), aspas triplas simples (""") ou aspas triplas duplas (""").
- Por serem sequências imutáveis, cada caractere de uma string ocupa uma posição específica na memória.
- Assim, é possível acessar caracteres individuais utilizando colchetes ([]) e o índice correspondente à posição desejada.

Strings

- Exemplos:

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 print(s[0])  
3 print(s[0:6])  
4 print(s[:6])  
5 print(s[-1])  
6 print(s[-12:-1])  
7 print(s[-12:])
```



```
lucas@lucas-Inspiron  
3 /home/lucas/Docume  
L  
Lógica  
Lógica  
!  
programação  
programação!
```

Strings

- Exemplos:

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 print(s[0])  
3 print(s[0:6])  
4 print(s[:6])  
5 print(s[-1])  
6 print(s[-12:-1])  
7 print(s[-12:])
```



```
lucas@lucas-Inspiron  
3 /home/lucas/Docume  
L  
Lógica  
Lógica  
!  
programação  
programação!
```

Fatiar (ou "slicing") uma string em Python é o processo de extrair uma subsequência de caracteres da string original. Isso é feito especificando um intervalo de índices.

Strings

- Como uma string é uma sequência de caracteres, ela pode ser percorrida utilizando laços de repetição, como for ou while.

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 i=0  
3 while i != len(s):  
4     print(s[i])  
5     i += 1
```

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 i=0  
3 while i != len(s):  
4     i += 1  
5     print(s[-i])
```

Strings

- Como uma string é uma sequência de caracteres, ela pode ser percorrida utilizando laços de repetição, como for ou while.

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 for i in range(len(s)):  
3     print(s[i])
```

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 for i in range(len(s)):  
3     print(s[-i-1])
```

Strings

- Como uma string é uma sequência de caracteres, ela pode ser percorrida utilizando laços de repetição, como for ou while.

```
main.py > ...  
1  s = "Lógica de programação"  
2  
3  for c in s:  
4      | print(c)  
5
```

Strings

- É possível verificar se um determinado caractere (ou até mesmo uma sequência de caracteres) está presente em uma string utilizando o operador `in`. Esse operador retorna um valor booleano (`True` ou `False`), indicando a presença ou ausência do elemento.

```
s = "Lógica de programação"
c = "L"

if c in s:
    print(f"O caractere {c} está presente na string {s}")
```


Strings

- Para verificar se um caractere (ou sequência de caracteres) não está presente em uma string, utiliza-se o operador `not in`. Ele retorna `True` se o elemento não for encontrado na string, e `False` caso contrário.

```
s = "Lógica de programação"
c = "Z"

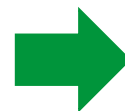
if c not in s:
    print(f"O caractere {c} não está presente na string {s}")
```

Strings

- Funções comuns para manipulação de strings em Python:
 - `upper()`: converte todos os caracteres da string para maiúsculos.
 - `lower()` ou `casefold()`: converte todos os caracteres da string para minúsculos.
 - `find()`: busca a posição da primeira ocorrência de um caractere (ou substring) na string; retorna -1 se não for encontrado.
 - `count()`: conta quantas vezes um determinado caractere (ou substring) aparece na string.
 - `split()`: divide a string em uma lista, utilizando como separador o espaço por padrão (ou outro delimitador, se especificado).
 - `len()`: retorna o número de caracteres da string.

Strings

```
main.py > ...  
1 s = 'Lógica de programação!'  
2 i=0  
3 print(s.upper())  
4 print(s.lower())  
5 print(s.find('!'))  
6 print(s.find('a'))  
7 print(s.find('de'))  
8 print(s.count('a'))  
9 print(s.split(' '))
```



```
lucas@lucas-Inspiron-5548:~/Docum  
3 /home/lucas/Documents/vscode-p  
LÓGICA DE PROGRAMAÇÃO!  
lógica de programação!  
21  
5  
7  
3  
['Lógica', 'de', 'programação!']
```

Strings

- Outras funções úteis para manipulação de strings em Python:
 - `capitalize()`: coloca a primeira letra da string em maiúscula e as demais em minúscula.
 - `format()`: insere valores em uma string de forma formatada, substituindo marcadores.
 - `isalnum()`: verifica se todos os caracteres da string são alfanuméricos (letras e números); letras com acento ou o caractere ç também são considerados letras.
 - `isalpha()`: verifica se todos os caracteres da string são apenas letras (sem números ou símbolos).
 - `isnumeric()`: verifica se todos os caracteres da string são apenas dígitos numéricos.

Strings

- Outras funções úteis para manipulação de strings em Python:
 - `replace()`: substitui todas as ocorrências de um trecho da string por outro.
 - `splitlines()`: divide a string em uma lista, separando-a onde houver quebras de linha (`\n`).
 - `startswith()`: verifica se a string começa com um determinado texto.
 - `strip()`: remove espaços em branco (ou caracteres especificados) do início e do fim da string.
 - `title()`: coloca a primeira letra de cada palavra da string em maiúscula.

Strings

- Para obter o ponto de código Unicode de um caractere, usamos a função `ord()`.

```
caractere = 'A'  
valor_ascii = ord(caractere)  
print(valor_ascii) # Saída: 65
```

- Para obter o caractere correspondente a um ponto de código Unicode, usamos a função `chr()`.

```
valor_ascii = 65  
caractere = chr(valor_ascii)  
print(caractere) # Saída: 'A'
```

Exercícios

1. Faça um programa que leia 2 strings e informe o conteúdo delas seguido do seu comprimento. Informe também se as duas strings possuem o mesmo comprimento e são iguais ou diferentes no conteúdo.
2. Faça um programa que permita ao usuário digitar o seu nome e em seguida mostre o nome do usuário de trás para frente utilizando somente letras maiúsculas. Dica: lembre-se que ao informar o nome, o usuário pode digitar letras maiúsculas ou minúsculas.

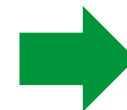
Exercícios

3. Faça um programa que solicite uma string ao usuário e em seguida a imprima em formato de escada.



```
lucas@lucas  
3 /home/luc  
L  
Lu  
Luc  
Luca  
Lucas
```

4. Altere o programa anterior de modo que a escada seja invertida.



```
lucas@luca  
3 /home/lu  
Lucas  
ucas  
cas  
as  
s
```


Exercícios

5. Um palíndromo é uma sequência de caracteres cuja leitura é idêntica se feita da direita para esquerda ou vice-versa. Por exemplo: OSSO e OVO são palíndromos. Em textos mais complexos os espaços e pontuação são ignorados. A frase SUBI NO ONIBUS é o exemplo de uma frase palíndroma onde os espaços foram ignorados. Faça um programa que leia uma sequência de caracteres, mostre-a e diga se é um palíndromo ou não.

Exercícios

6. Faça uma função que recebe uma string que representa uma cadeia de DNA e gera a cadeia complementar. A entrada e saída de dados deve ser feita pelo programa principal.
Exemplo: Entrada: AATCTGCAC
Saída: TTAGACGTG
7. Faça um programa que leia uma data de nascimento no formato dd/mm/aaaa e imprima a data com o mês escrito por extenso.
Exemplo: Data = 20/02/1995
Resultado gerado pelo programa: Você nasceu em 20 de fevereiro de 1995

Exercícios

8. Escreva um programa que leia duas strings. Verifique se a segunda ocorre dentro da primeira e imprima a posição de início.

Exemplo: 1ª string: AABBEFAATT

2ª string: BE

Resultado: BE encontrado na posição 3 de AABBEFAATT

9. Escreva um programa que leia duas strings e gere uma terceira com os caracteres comuns às duas strings lidas.

Exemplo: 1ª string: AAACCTBF

2ª string: CBT

Resultado: CBT

A ordem dos caracteres da string gerada não é importante, mas deve conter todas as letras comuns a ambas.

Exercícios

12. Número por extenso. Escreva um programa que solicite ao usuário a digitação de um número até 99 e imprima-o na tela por extenso.
13. Faça um programa que leia uma palavra e some 1 no valor Unicode de cada caractere da palavra. Imprima a string resultante.
Dica: O Python disponibiliza 2 funções que são bastante uteis quando estamos trabalhando com o sistema Unicode. A primeira é a função `ord()` , que recebe uma letra como parâmetro e retorna o código Unicode da mesma. A segunda função, é a `chr()`, onde passamos o código Unicode e nos é retornado a respectiva letra.
Tabela ASCII (subconjunto da tabela Unicode):
<https://www.ime.usp.br/~pf/algoritmos/apend/ascii.html>

Exercícios

14. Faça um programa que solicite ao usuário uma string e modifique a string para que todos os caracteres fiquem em maiúsculas. Obs: Não utilize a função `upper()`.
Obs: Utilize a tabela ASCII.
15. Faça um programa em que troque todas as ocorrências de uma letra L1 pela letra L2 e da L2 pela L1 em uma string. A string e as letras L1 e L2 devem ser fornecidas pelo usuário.
Obs: Não utilize a função `replace()`.

Dúvidas



LÓGICA E LINGUAGEM DE PROGRAMAÇÃO

Curso Técnico Subsequente em Informática
Lucas Sampaio Leite

