Disciplina Lógica de Programação

Strings em Python

Téc. Informática Integrado ao Ensino Médio - 2018

Prof. Dr. Paulo César Rodacki Gomes
paulo.gomes@ifc.edu.br

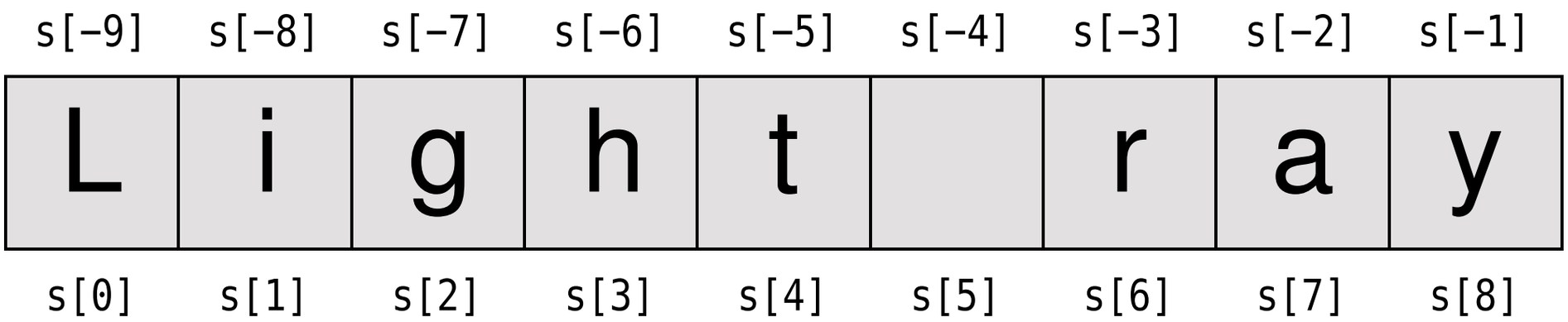
INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Strings

- Podem ser consideradas um tipo de dados composto pois uma string é uma seqüência de elementos menores chamados caracteres

- Podem ser indexadas pelo operador [ ]

- exemplo: s = “Light ray”



Operações

- o operador de adição funciona com strings:
- Exemplo:

```
fruta = " banana"
tipo  = "bolo de"
print(tipo + fruta)
```

- Imprime: bolo de banana

Operações

- o operador * funciona para fazer repetições:
- Exemplo:

```
print("Vai "*6)
nome = "Brasil"
print(nome * 3)
print(nome + " Vai" * 3)
print((nome + " Vai") * 3)
```

- Imprime:

```
Vai Vai Vai Vai Vai Vai
BrasilBrasilBrasil
Brasil Vai Vai Vai
Brasil VaiBrasil VaiBrasil Vai
```

Indexação

- o operador de adição funciona com strings:
- Exemplo:

```
escola = "Instituto Federal Catarinense"
m = escola[2]
print(m)
ultimo = escola[-1]
print(ultimo)
```

- Imprime: **s** e depois **e**

Métodos de String

- em Python, strings são objetos (no próximo ano você vai saber mais sobre objetos), por isso elas tem atributos e métodos

- Exemplo:

```
ss = "Hello, World"
print(ss.upper())
tt = ss.lower()
print(tt)
```

- Imprime:

```
HELLO, WORLD
hello, world
```

Método	Parâmetros	Descrição
upper	nenhum	Retorna um string todo em maiúsculas
lower	nenhum	Retorna um string todo em minúsculas
capitalize	nenhum	Retorna um string com o primeiro caractere em maiúscula, e o resto em minúsculas
strip	nenhum	Retorna um string removendo caracteres em branco do início e do fim
lstrip	nenhum	Retorna um string removendo caracteres em brando do início
rstrip	nenhum	Retorna um string removendo caracteres em brando do fim
count	item	Retorna o número de ocorrências de item
replace	old, new	Substitui todas as ocorrências do substring old por new
center	largura	Retorna um string centrado em um campo de tamanho largura
ljust	largura	Retorna um string justificado à esquerda em um campo de tamanho largura
rjust	largura	Retorna um string justificado à direita em um campo de tamanho largura
find	item	Retorna o índice mais à esquerda onde o substring item é encontrado
rfind	item	Retorna o índice mais à direita onde o substring item é encontrado
index	item	Como find, mas causa um erro em tempo de execução caso item não seja encontrado
rindex	item	Como rfind, mas causa um erro em tempo de execução caso item não seja encontrado

Métodos de String

- Exemplo:

```
ss = "    Hello, World    "
els = ss.count("l")
print(els)
print("***"+ss.strip()+"***")
print("***"+ss.lstrip()+"***")
print("***"+ss.rstrip()+"***")
news = ss.replace("o", "***")
print(news)
```

- Imprime:

```
3
***Hello, World***
***Hello, World    ***
***    Hello, World***
    Hell***, W***rld
```

- Exemplo:

```
food = "banana bread"
print(food.capitalize())
print("*"+food.center(25)+"*")
print("*"+food.ljust(25)+"*")
print("*" +food.rjust(25)+"*")
print(food.find("e"))
print(food.find("na"))
print(food.find("b"))
print(food.rfind("e"))
print(food.rfind("na"))
print(food.rfind("b"))
print(food.index("e"))
```

Imprime:

```
Banana bread
*       banana bread        *
*banana bread               *
*                banana bread*
9
2
0
9
4
7
9
```

Comprimento

- A função len, quando aplicada a um string, retorna o número de caracteres numa string (ou seja, o seu comprimento):
- Exemplo:

```
fruta = "Banana"
print(len(fruta))
```

- Imprime: **6**

Fatiamento

- Um substring de um string é chamado de fatia (do inglês slice). Selecionar uma fatia é semelhante a selecionar um caractere

- O operador de fatiamento possui três sintaxes:

```
seq[start]
seq[start:end]
seq[start:end:step]
```

- start, end e step precisam ser números inteiros e eventualmente podem ser suprimidos

Fatiamento

- Exemplo:

```
singers = "Peter, Paul, and Mary"
print(singers[0:5])
print(singers[7:11])
print(singers[17:21])
```

- Imprime:

```
Peter
Paul
Mary
```

Fatiamento

- Se você omitir o primeiro índice (antes dos dois pontos), a fatia começa no início da cadeia. Se você omitir o segundo índice, a fatia vai até o fim da cadeia

- Exemplo:

```
fruta = "banana"
print(fruta[:3])
print(fruta[3:])
```

- Imprime:

```
ban
ana
```

Comparação de Strings

- Os operadores de comparação também funcionam com strings. Para ver se dois strings são iguais, basta escrever uma expressão booleana usando o operador de igualdade.

- Exemplo:

```
word = "banana"
if word == "banana":
    print("Yes, we have bananas!")
else:
    print("Yes, we have NO bananas!")
```

- Imprime:

```
Yes, we have bananas!
```

Comparação de Strings

- Outras operações de comparação são úteis para colocar palavras em ordem *lexicográfica*

- Exemplo:

```
word = "zebra"
if word < "banana":
    print("Your word, "+word +", comes before banana.")
elif word > "banana":
    print("Your word, "+word+", comes after banana.")
else:
    print("Yes, we have no bananas!")
```

- Imprime:

```
Your word, zebra, comes after banana.
```

Strings são imutáveis

- você não tem permissão para modificar os caracteres individuais na string

- Exemplo:

```
conversa = "Ola, mundo!"
conversa[0] = 'B'   # ERRO!!!!!!
print(conversa)
```

Strings são imutáveis

- Consertando o código anterior…
- Exemplo:

```
conversa = "Ola, mundo!"
nova_conversa = 'B' + conversa[1:]
print(nova_conversa)
print(conversa)
```

- Imprime:

```
Bla, mundo!
Ola, mundo!
```

Varrendo com for

- Como um string é simplesmente uma sequência de caracteres, o laço for itera sobre cada caractere automaticamente

- Exemplo:

```
for um_char in "Instituto":
    print(um_char)
```

Imprime:

```
I
n
s
t
i
t
u
t
o
```

Varrendo com for e range

- Também é possível utilizar a função range para gerar sistematicamente os índices dos caracteres

- Exemplo:

```
fruta = "apple"
for idx in range(5):
    currentChar = fruta[idx]
    print(currentChar)
```

Imprime:

```
a
p
p
l
e
```

Varrendo com while

- O laço while também pode controlar a geração dos valores de índices. Lembre-se que o programador é responsável por configurar a condição inicial, certificando-se que a condição é correta e certificando-se de que algo muda dentro do corpo para garantir que a condição se tornará falsa.

- Exemplo:

```
fruta = "apple"
position = 0
while position < len(fruta):
    print(fruta[position])
    position = position + 1
```

Imprime:

```
a
p
p
l
e
```

Operadores in e not in

- O operador **in** testa se um string é uma substring de outra
- Exemplo:

```
print('p' in 'apple')
print('i' in 'apple')
print('ap' in 'apple')
print('pa' in 'apple')
```

Imprime:

```
True
False
True
False
```

- O operador **not in** retorna o resultado lógico oposto de **in**
- Exemplo:

```
print('x' not in 'apple')
```

Imprime:

```
True
```