Strings

Programação I

2016.2017

Teresa Gonçalves tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Reminder...



Como aprender?

Estudar, estudar, estudar...

Praticar, praticar, praticar...

Cometer erros, cometer erros, cometer erros...

Aprender com os erros, aprender com os erros, aprender com os erros ...

Sumário

Propriedades, operadores e indexação

Iteração

Métodos

Símbolos especiais



String

Representação de texto Sequência de caracteres

Tamanho da sequência

Função len()

Propriedades

Sequencial

Relação de ordem entre elementos

Indexável

acesso direto a cada elemento através do índice

Imutável

impossível alterar os seus elementos

Operadores básicos

Concatenação

```
>>> a='um' + 'Teste'
>>> print(a)
umTeste
```

Repetição

```
>>> a='Évora' * 3
>>> print(a)
ÉvoraÉvoraÉvora
```

Indexação

Acesso direto a um caracter da string

Operador []

Índice

Pode ser uma expressão que contém variáveis e operadores... mas tem de ser um inteiro

```
1º elemento: 0
```

Último elemento: len()-1

```
>>> letter=fruit[1]
>>> letter
'a'
```



Índices negativos

Contam do fim para o início

```
>>> fruit ='banana'
>>> lenght = len(fruit)
>>> last=[lenght-1]
>>> fruit[-1]
'a'
>>> fruit[-2]
'n'
```



Segmento da string

Operador [m:n]

Devolve os elementos da sequência da posição m até à posição n

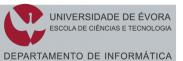
Inclui m mas exclui n

Omissão

```
1º índice – segmento começa no início da sequência
```

2º índice – segmento termina no fim da sequência

```
>>> s = 'Monty Python'
>>> s[1:5]
'onthy'
>>> s[6:]
'Python'
```



Operador in

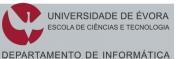
str1 in str2

Operador binário booleano

Operação entre 2 strings

Retorna True se a primeira string é substring da segunda

```
>>> s="Rua Romao Ramalho, n.1, 7000 Evora"
>>> "Evora" in s
True
>>> "evora" in s
False
>>> "Lisboa" not in s
True
```



Imutabilidade

Não é possível alterar uma string existente

```
>>> greeting='hello world!'
>>> greeting[0] = 'j'
TypeError: 'str' object does not support item assignment
```

Como fazer?

```
Criar uma nova string, variante da original
```

```
>>> greeting='hello world!'
>>> new_greeting = 'j' + greeting[1:]
>>> new_greeting
'jello world!'
```



Iteração

Iteração sobre strings

Iterar na sequência

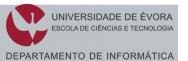
```
for char in s:
    print( char )
```

Iterar sobre os índices (for)

```
for indice in range(len(s)):
    print( s[indice] )
```

Iterar sobre os índices (while)

```
i=0
while i<len(s):
    print( s[i] )
    i = i+1</pre>
```



Comparação

Igualdade

==

Desigualdade

Ordenação baseada no código ASCII dos caracteres

dígitos < maiusculas < minúsculas

ord(l) devolve o código da letra l





Método

Função aplicada a um objeto Utilização

```
var.metodo()
var.metodo(argumentos)
```

Métodos sobre strings

```
upper(), lower()
isspace(), isupper(), islower(), isdigit(), isalpha()
find(), count()
format(), split(), replace(), join()
```

Conversão: upper(), lower()

s.upper()

Devolve uma nova string cujo valor é a conversão de s para maiúsculas

s.lower()

Devolve uma nova string cujo valor é a conversão de s para minúsculas

```
>>> "um Exemplo".upper()
'UM EXEMPLO'
>>> s = 'Ponte Vasco da Gama'
>>> s.lower()
'ponte vasco da gama'
```



Teste: is...()

s.isupper()

True se é formada por maiúsculas

s.islower()

True se é formada por minúsculas

s.isspace()

True se é formada por espaços e/ou tabulações

s.isdigit()

True se inclui apenas algarismos

s.isalpha()

True se inclui apenas letras

s.alnum()

True se inclui apenas caracteres alfanuméricos



Procura: find()

s.find(str, start, end)

Devolve o índice da primeira ocorrência da string str na string s, ou -1 se não encontrar

start, end: índices opcionais de limite para a procura

```
>>> s='alentejo'
>>> s.find('tejo')
4
>>> s.find('tejo',1,5)
-1
>>> s.find('e',3)
5
```



Contagem: count()

s.count(str, start, end)

Devolve a contagem de ocorrências da string str na string s start, end: índices opcionais de limite para a procura

```
>>> s= "europa e america atravessam uma crise
economica"
>>> s.count('ca')
2
>>> s.count('a',1,10)
2
>>> s.count('a',30)
2
```

startswith(), endswith()

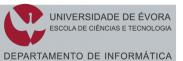
s.startswith(prefix)

Verifica se s inicia com o valor de prefix

s.endswith(sufix)

Verifica se s termina com o valor de sufix

```
>>> s= 'um exemplo'
>>> s.startswith('um')
True
>>> s.endswith('a')
False
>>> 'Outro'.endswith('o')
True
```



format()

s.format(args)

Permite formatar uma string constituida por partes constantes e partes variáveis

Parte estática

é escrita em s formando um padrão

Inclui anotações, denotadas por {}, que indicam onde colocar as partes variáveis

As anotações podem ser identificadas por índice ou nome

Argumentos

Expressões cujo resultado da avaliação é inserido em s na anotação correta.



format()

```
>>> "1+1 is {0}".format(1+1)
'1+1 is 2'
>>> "1+1 is {a}, x={b} ".format(a=2,b=8)
'1+1 is 2, x=8'
```

split()

s.split(sep, maxsplit)

Divide a string s e devolve uma lista com as partes sep – string usada como separador. Opcional. Valor por omissão: espaço maxsplit – nº máximo de divisões. Opcional

```
>>> s = "uma frase com varias palavras"
>>> s.split()
['uma', 'frase', 'com', 'varias', 'palavras']
>>> s.split('a',1)
['um', ' frase com varias palavras']
```



strip()

s.strip(chars)

Devolve uma cópia de s, removendo as ocorrências de caracteres em chars que surgirem no início ou no fim

chars – Opcional. Valor por omissão: espaços e tabulações

Exemplo

```
>>> ' spacious '.strip()
'spacious'
>>> 'www.example.com'.strip('cmowz.')
'example'
```

Funções relacionadas

lstrip(), rstrip()



replace()

s.replace(old, new)

Devolve cópia de s, substituindo as ocorrências de old por new

```
>>> s = "acto de escrever de facto a palavra acto"
>>> s.replace("acto","ato")
'ato de escrever de fato a palavra ato'
```

join()

s.join(sequencia)

Devolve uma nova string formada pela junção de cada elemento da sequência, usando s como separador

```
>>> '--'.join(['a','b','c'])
'a--b--c'
>>> ' '.join('isto')
'i s t o'
```





Símbolos especiais

Como incluir símbolos especiais

Exemplos: Apóstrofo, Mudança de linha, Tabulação

Usar um escape character: \

Apóstrofo: \'

Mudança de linha: \n

Tabulação: \t

O caracter \ indica que o caracter seguinte tem uma leitura especial!



Símbolos especiais

Exemplo 1

```
>>> s = 'duas\n linhas'
>>> print(s)
duas
 linhas
>>> s1 = "Évora"
>>> s2= "\tÉvora"
>>> print(s1)
Évora
>>> print(s2)
        Évora
```

Continuação de string

Caracter \

String literal que ocupa diversas linhas

... mas o conteúdo apenas uma!

```
>>> s = 'Esta é uma frase muito comprida. Para tornar a \
leitura mais fácil, está dividida em várias linhas através \
o carácter backslash \\. Este caracter indica que a linha \
seguinte é continuação da anterior'
>>> print(s)
Esta é uma frase muito comprida. Para tornar a leitura mais fácil, está dividida em várias linhas através do carácter backslash \. Este caracter indica que a linha seguinte é continuação da anterior
```



Operador %

Formatação de strings

'raio ' + str(valor + ' metros'

s.format()

Operador %

formato % valores

formato

string com modelo. Inclui texto com a posição e formatação pretendida para cada valor.

valores

Tuplo com expressões a serem avaliadas e colocadas na posição correspondente no modelo indicado em formato



Operador %

```
>>> r=3
>>> area= math.pi*r**2
>>> s= 'um circulo de raio %d tem area %f' % (r, area)
>>> print( s )
um circulo de raio 3 tem area 28.274334
```

Formato

% <flag> <largura mínima> <precisão> <conversão>

<flag>: o que fazer com a largura mínima? (opcional)

Opcional

0: preencher com zeros

+: apresentar sinal explícito

cisão>: .n, com n casas decimais

opcional

<conversão>: tipo pretendido para a conversão

d – inteiro s – string

o – base 8

f – float c – caracter

e – notação científica



Exemplos

Mostrar valor inteiro e com 2 casas decimais

```
>>> alunos = 11; soma = 147
>>> media = soma/alunos
>>> msg = '%d alunos tiveram media=%.2f' % (alunos, media)
>>> print(msg)
11 alunos tiveram media=13.36
```

Padding

```
>>> msg = '%05d alunos tiveram media=%e' % (alunos, media)
00011 alunos tiveram media=1.336364e+01
>>> msg = '%5d alunos tiveram media=%.2f' % (alunos, media)
11 alunos tiveram media=13.36
```



Exercício: minúsculas na string?

```
def any_lower1(s):
                                def any lower3(s):
    for c in s:
                                     for c in s:
        if 'c'.islower():
                                         if c.islower():
            return True
                                             return True
        else:
                                         else:
            return False
                                             return False
                                def any lower4(s):
                                     for c in s:
def any_lower2(s):
    for c in s:
                                         if not c.islower():
        flag = c.islower()
                                             return False
    return flag
                                     return True
```

Exercícios

Exercício 1

```
>>> s1 = 'Uma nova frase'
>>> s2 = s1[:4] + s1[4:]
>>> s1 == s2
```

Exercício 2

```
>>> s3 = s1[:-1]
>>> print( s3 )
```

Exercício 3

Como comparar 2 strings sem considerar a ordem ASCII dos caracteres?



Exercícios

Cifra ROT13

Codifica cada letra pela que está 13 posições depois

Função codificadora

Assumir que existem apenas caracteres minúsculos/maiúsculos

rot13('hello') == 'uryyb'

Qual a função para descodificar?