ENTIDADE FORMADORA CERTIFICADA
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL, IP

PPT 08 – Fundamentos de Python (UFCD 10793)

Sandra Liliana Meira de Oliveira

PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU

Strings

Strings - Declaração

```
a ="Olá Mundo!!"

b='Olá Mundo'

c="""Programar em Python é como
andar de bicicleta ... depois de
aprender não se esquece"""

d='''Programar em Python é como
andar de bicicleta ... depois de
aprender não se esquece'''
```

Strings – intervalo de carateres

```
a ="Olá Mundo!!"


print(a[2])          #imprime o carater na posição 2

print(a[4: 7])       #imprime desde o carater na posição 4 (incluido) até ao carater
                     #da posição 7 (não incluido)

print(a[4:])         #imprime desde o carater da posição 4 até ao final da string

print(a[:3])         #imprime desde inicio até ao carater da posição 3 (não incluido)

print(a[-5: -2])     #imprime desde o carater na posição 5 a contar do fim (incluido)
                     #até ao carater da posição 2 a contar do fim (não incluido)
```

Strings - Modificar

```
a =" Olá, Mundo!! "

print(a.upper()) # Coloca a String com letras maiusculas

print(a.lower()) # Coloca a String com letras minusculas

print(a.strip()) # Remove qualquer espaço no inicio e no fim da string

print(a.split(",")) # Devolve lista dividida pelo carater "," ['Olá', 'Mundo!!']
```

Nota: Existem ou comandos modificam e testam strings.

Strings – Juntar strings

```
nome="Ana"
idade=23

print("O meu nome é", nome, "e tenho", idade, "anos de idade")

print("O meu nome é {} e tenho {} anos de idade".format(nome, idade))

print(f"o meu nome é {nome} e tenho {idade} anos de idade")
```

Strings – Carateres especiais

Carateres especiais	Descrição
\'	Plica
\\	barra
\n	Nova linha
\t	Tab
\b	Backspace

Strings

- Uma String é uma sequência de caracteres.
- Em Python, Strings são representadas como listas imutáveis de caracteres.
- Podemos representar uma String como uma sequência de caracteres entre aspas simples (') ou aspas duplas (").
- Exemplo:

```
1 msg = " hello world"
2 print(msg)
3 # hello world
```

Caracteres Especiais

- O seguinte trecho de código apresenta erros.

```
print("Respondeu "SIM".") #
SyntaxError: invalid syntax
print("\")
# SyntaxError: EOL while scanning string literal
```

- Isso acontece porque " e \ são caracteres reservados da linguagem.
- Para representar os caracteres " e \ precisamos utilizar o seguinte código.

```
print("Respondeu \"SIM\".") #
Você respondeu " SIM".
print("\\")
# \
```

Aceder Elementos de uma String

```
msg = " hello world"
print(msg[0])
# h  print(
msg[1])  # e
print(msg[-1])
# d  print(
msg[-5])    # w
print(msg[12])
# IndexError: string index out of  range
```

Aceder a Elementos de uma String

- Como Strings são listas imutáveis (assim como as tuplas), não é possível alterar uma posição da String.

```
1 msg = " hello world"
2 msg[0] = "y"
3 # TypeError: 'str' object does not support item assignment
```

Aceder Elementos de uma String

```
msg = " hello world"
print(msg[3:8])
# lo wo  print(
msg[:5])  #
hello  print(
msg[6:])  #
world  print(
msg[::2])   #
hlowrd
print(msg[::-1])
# dlrow olleh
```

Operações, Funções e Métodos

Concatenação de Strings

- O operador `+` concatena duas Strings.

```
msg = " hello"
msg2 = "y" + msg[1:] + "w"
print(msg2)
# yellow
```

- O operador `*` replica uma String.

```
s = " abc"
print(s  * 3)
# abcabcabc
```

Tamanho de uma String

- A função `len` retorna o tamanho (quantidade de caracteres) de uma String.

```
msg = " hello"
print(len(msg))
# 5
msg2 = " Hello World"
print(len(msg2))
# 11
msg3 = "Hello\nWorld"
print(len(msg3))
# 11

```

- Observe que qualquer tipo de caractere é contado pela função `len`, inclusive espaços, quebra de linhas ou tabulações.

Comparação de Strings

- O operador `==` verifica se duas Strings são iguais.
- O operador `!=` verifica se duas Strings são diferentes.

```
a = " Python"
b = "Py" + " thon"
c = "p" + " ython"
print(a  == b)
# True
print(a  == c)
# False
print(b  != c)
# True
```

- O operador `in` verifica se uma String é parte de outra String.

```
print("thon" in  "Python")
# True
print("thor" in  "Python")
# False
```

Comparação de Strings

- O método `startswith` verifica se a String recebida como parâmetro é um prefixo da String base.

```
msg = " Hello World"  print(
msg.startswith("Hello"))
#True
print(msg.startswith(" World"))
#False
```

Search uma String

- O método `index` retorna a primeira posição em que uma String fornecida como parâmetro ocorre na String base.
- Se a String fornecida como parâmetro não está contida na String base, então é gerado um erro (similar ao que ocorre com listas).

```
1 bond = "My name is Bond , James  Bond"
2 print(bond.index("Bond"))
3 # 11
4 msg = " Hello World"
5 print(msg.index("World"))
6 # 6
7 print(msg.index("Bond"))
8 # ValueError: substring not found
```

Pesquisar uma String

- O método `find` também retorna a primeira posição em que uma String fornecida como parâmetro ocorre na String base.
- Se a String fornecida como parâmetro não está contida na String base, então é retornado o valor `-1`.

```
1 bond = "My name is Bond , James  Bond"
2 print(bond.find("Bond"))
3 # 11
4 msg = " Hello World"
5 print(msg.find("World"))
6 # 6
7 print(msg.find("Bond"))
8 # -1
```

Manipulação de Strings

- O método `strip` remove todos os espaços em branco (incluindo quebras de linhas e tabulações) no início e no fim da String.

```
1 msg = " \n   Hello World \t"
2 print(msg.strip())
3 # Hello World
```

Manipulação de Strings

- O método `split` divide uma String em uma lista de acordo com um padrão de caracteres (separador).
- Por padrão, o separador é igual a qualquer sequência de espaços em branco (incluindo quebras de linhas e tabulações).

```
1 str1 = "   MC102   Algoritmos\t\tProgramação\nComputadores    "
2 dados = str1.split()
3 print(dados)
4 # ['MC102', 'Algoritmos', 'Programação','Computadores']
5 str2 = "abacaxi, banana, caqui, damasco"
6 frutas = str2.split(", ")
7 print(frutas)
8 # [' abacaxi', 'banana', 'caqui', 'damasco']
```

Lendo Múltiplos Valores

- Por exemplo, o método `split` pode ser usado para separar múltiplos valores lidos numa única linha.

```
# Lendo duas strings separadas por um espaço
s1 , s2 = input().split()
# Lendo três números inteiros separados por espaços
a, b, c = [ int(i) for i in input(). split()]
# Lendo múltiplos números separados por espaços
numeros = [ float(i) for i in input().split()]
```

Manipulação de Strings

- O método `join` junta uma lista de Strings usando a String base como concatenador.

```
frutas = ['abacaxi', 'banana', 'caqui', 'damasco']
txt = ", ".join(frutas)
print(txt)
# abacaxi, banana, caqui, damasco
```

- A função `list()` pode ser utilizada para transformar uma String numa lista de caracteres.

```
str = " aeiou"
lista = list(str)
print(lista)
# ['a', 'e', 'i', 'o', 'u']
```

Manipulação de Strings

- O método `replace` cria uma nova String onde todas as ocorrências de um padrão de caracteres numa String dada são trocadas por outro.

```
1  x = " Algoritmos e Programação de Computadores"
2  y = x. replace("a", "_")
3  print (y)
4  # Algoritmos e Progr_m_ção de Comput_dores
5  y = x.replace("Algoritmos", "$" * len("Algoritmos"))
6  print (y)
7  # $$$$$$$$$$ e Programação de Computadores
8  x = "a,b,c,d,e"
9  y = x. replace(",", "")
10 print (y)
11 # abcde
```

Outros Métodos

- `capitalize()`: converte o primeiro caractere para maiúsculo.

```
print("meu teste".capitalize())
# Meu teste
```

- `lower()`: converte a String para letras minúsculas.

```
print("Meu TESTE".lower())
# meu teste
```

- `upper()`: converte a String para letras maiúsculas.

```
print("mEU tESte".upper())
# MEU TESTE
```

Outros Métodos

- `isnumeric()`: testa se todos os caracteres são dígitos.

```
print("1234".isnumeric())
# True
print("teste123".isnumeric())
# False
```

- `isalpha()`: testa se todos os caracteres são letras.

```
print("MeuTeste".isalpha())
# True  print("teste123".
isalpha())  # False

```

- `isalnum()`: testa se todos os caracteres são letras ou dígitos.

```
print("teste123".isalnum())
# True
print("Meu teste".isalnum())
# False
```

Exemplo

- Exemplo:

```
1  sc = input(" Introduza uma sequência de caracteres ")
2
3  if sc.isalpha():
4      print(sc, " possui apenas letras")
5  elif sc.isnumeric():
6      print(sc, " possui apenas dígitos")
7  elif sc.isalnum():
8      print(sc, " possui letras e dígitos")
9  else:
10     print(sc, " não possui apenas letras e dígitos")
```

Ciclos e Strings

- Podemos utilizar o comando `for` para percorrer uma String.
- Exemplo:

```
s = " abc"
for c in s:
  print(c)
# a
# b
# c
```

- Outro exemplo:

```
for c in " Algoritmos":
    if c in "AEIOUaeiou":
     print("A String possui a vogal:", c)
# A String possui a vogal: A
# A String possui a vogal: o
# A String possui a vogal: i
# A String possui a vogal: o
```

Exercícios

Exercícios

1. Escreva um programa que, dada uma sequência de números inteiros (todos fornecidos na mesma linha, separados por espaços), imprima a média desses números.
2. Escreva um programa que, dada uma String representando um texto, imprima o número de palavras existentes. Observação: você deve remover os sinais de pontuação (".", ",", ":", ";", "!" e "?") antes de realizar a contagem das palavras.

Exercício 1

1. Escreva um programa que, dada uma sequência de números inteiros (todos fornecidos na mesma linha, separados por espaços), imprima a média desses números.

```
texto = input(" Entre com uma sequência de números: ")
numeros =texto. split()
soma = 0

for n in numeros:
   soma = soma + int(n)
media = soma / len( numeros)

print("A média é: ", format(media , ".2f"))
```

Exercício 2

2. Escreva um programa que, dada uma String representando um texto, imprima o número de palavras existentes. Observação: você deve remover os sinais de pontuação (“.”, “,”, “:”, “;”, “!” e“?”) antes de realizar a contagem das palavras.

```
1 texto = input(" Introduza um texto: ")
2 pontuacao = [".", ",", ":", ";", "!", "?"]
3
4 # remove os sinais de pontuação
5 for p in pontuacao:
6   texto = texto. replace(p, " ")
7
8 numPalavras = len( texto. split())
9 print(" Número de palavras:", numPalavras)
```