



PONTIFÍCA UNIVERSIDADE CATÓLICA DE CAMPINAS

Escola Politécnica - Curso de Engenharia de Software

12413 - ALGORITMOS DE PROGRAMAÇÃO, PROJETOS E COMPUTAÇÃO

ASSUNTOS:

- correção de exercício
- estrutura de dados - string;

Profa. Angela de Mendonça Engelbrecht

angel@puc-campinas.edu.br

profa.angela@gmail.com

File Edit Format Run Options Window Help

```
#correção do exercício
#Construir um programa que imprime uma tabela
#com valores em Fahrenheit (de -50o até 50o,
#de 5 em 5 graus) e o respectivos valores em
#Celsius. Lembrando:  $C = 5/9 * (F - 32)$ 
```

```
print('    <<< CONVERSION DE F° PARA C° >>>')
print('='*30, '\n')
print('    F°    |    C°')
print('-'*30, '\n')
```

```
for F in range(-50, 51, 5):
    C = 5/9*(F-32)
    print(f'{F:5d}    |    {C:6.2f}')
```

```
= RESTART: C:/Users/profa/OneDrive/Área de Trabalho/
<<< CONVERSION DE F° PARA C° >>>
```

```
=====
```

```
    F°    |    C°
-----
```

```
-50    |    -45.56
-45    |    -42.78
-40    |    -40.00
-35    |    -37.22
-30    |    -34.44
-25    |    -31.67
-20    |    -28.89
-15    |    -26.11
-10    |    -23.33
-5     |    -20.56
0      |    -17.78
5      |    -15.00
10     |    -12.22
15     |     -9.44
20     |     -6.67
25     |     -3.89
30     |     -1.11
35     |      1.67
40     |      4.44
45     |      7.22
50     |     10.00
```

```
>>>
```

Usando for/range:

Escreva um programa que calcule e escreva o valor de S:

$$S = \frac{1}{1} - \frac{2}{4} + \frac{3}{9} - \frac{4}{16} + \dots - \frac{10}{100}$$

```
File Edit Format Run Options Window Help
'''Escreva um programa que calcule
e escreva o valor de S:
s= 1/1-2/4+3/9-4/16+...- 10/100
'''

S = 0
for numerador in range(1,11):
    fracao = numerador / numerador ** 2
    if numerador%2==0:
        S = S - fracao
    else:
        S = S + fracao
print(f'Resultado = {S:.2f}')
```

```
coes4.py
Resultado = 0.65
>>>
```

STRING – um caractere ou uma cadeia de caracteres

File Edit Format Run Options Window Help

```
''' string '''
```

```
nome_produto = 'cadeira'
```

```
preço = 250.00
```

```
quantidade = 20
```

```
frase = f'''No estoque, há {quantidade} {nome_produto}s,  
com preço R$ {preço:.2f} cada uma'''
```

```
No estoque, há 20 cadeiras,  
com preço R$ 250.00 cada uma
```

STRING – um caractere ou uma cadeia de caracteres

tipo: *str* de *string*

Representação: letra – maiúsculas e minúsculas, **dígitos**, **símbolos**;

Algumas operações:

concatenação (+): unir strings formando outras strings -> **s1 + s2**

duplicação (*): **s * n** -> duplicar **s**, **n** vezes , na própria **s**

len (s) - retorna o **comprimento** da string **s**

s1 in s2 : retorna **True** se **s1** está **contida** em **s2** e **False** caso contrário

s1 not in s2: retorna **True** se **s1** **não** estiver **contida** em **s2** e **False** caso contrário

Representação Interna de uma string:

Considere o exemplo de string:

```
>>> frase = 'Bom dia! Hoje é quarta-feira'
>>> frase
'Bom dia! Hoje é quarta-feira'
>>> 'a' in frase
True
>>> 'A' in frase
False
```

```
>>> len(frase)
28
```

frase

B	o	m		d	i	a	!		H	o	j	e		é		q	u	a	r	t	a	-	f	e	i	r	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

índices

uma *string* é representa: em sequência *indexada*, começando do **ZERO**

Representação Interna de uma string:

```
>>> len(frase)
28
```

frase

B	o	m		d	i	a	!		H	o	j	e		é		q	u	a	r	t	a	-	f	e	i	r	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

índices

Sabendo que a representação de uma string, possui **índices** -> podemos acessar cada um dos caracteres pelo seu respectivo **índice**:

```
>>>
>>> frase[0]
'B'
>>> frase[7]
'!'
>>> frase[13]
','
>>> frase[27]
'a'
>>> frase[28]
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    frase[28]
IndexError: string index out of range
>>>
```

o intervalo do índice de uma string:
[0,len(str)-1]

Fatias de uma string - *slices*

frase

B	o	m		d	i	a	!		H	o	j	e		é		q	u	a	r	t	a	-	f	e	i	r	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

índices

```
>>> #intervalo [0:5]==> posição [0] até [5] sem incluí-la
>>> frase[0:5]
'Bom d'
>>>
```

```
>>> #[3:20]==> de [3] até [20] sem inclui-la
>>> frase[3:20]
' dia! Hoje é quar'
>>>
```

```
>>> #omitir um dos números: desde o início ou até o fim
>>> frase[:5]
'Bom d'
>>> frase[20:]
'ta-feira'
```

```
>>> #omitindo ambos ==> do início até o fim
>>> frase[:]
'Bom dia! Hoje é quarta-feira'
>>>
```


frase

B	o	m		d	i	a	!		H	o	j	e		é		q	u	a	r	t	a	-	f	e	i	r	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

índices

sentido do intervalo

valores podem ser negativos – contagem do final para o início

```
>>> #negativos para contar do final para o começo
>>> #-1: último caractere
>>> #-2: penúltimo e assim por diante
>>> frase[-1:]
'a'
>>> frase[-2:]
'ra'
```

```
>>> #[-10:20] ==> [18:20]
>>> frase[-10:20]
'ar'
>>> frase[18:20]
'ar'
```

```
>>> #o que ele interpreta se pedirmos
>>> # [-2:20] ==> do penúltimo [26]--> [20]
>>> #contrário ao sentido --->
>>> frase[-2:20]
''
```

nada

Característica da string:

string: imutável

```
>>> frase
'Bom dia! Hoje é quarta-feira'
>>> frase[7]
'!'
>>> #quero mudar '!' por uma vírgula, por exemplo
>>> frase[7] = ','
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    frase[7] = ','
TypeError: 'str' object does not support item assignment
>>> |
```

Se for necessário, numa aplicação, efetuar **mudanças**, poderemos usar **outro tipo** de dado, para um conjunto de caracteres, e que poderá ser mudado - **mutável**

Se podemos ter **acesso** aos **caracteres** de uma **string** usando seu **índice**, poderia fazer um programa que **percorre** a string e **conta** quantos **espaços** em branco a **string** possui:

File Edit Format Run Options Window Help

```
''' string '''
```

```
frase = 'Bom dia! Hoje é quarta-feira.'
```

```
index = 0
```

```
conta = 0
```

```
while index < len(frase):
```

```
    if frase[index] == ' ':
```

```
        conta+=1
```

```
    index+=1
```

```
print(f'Total de espaços: {conta}' )
```

Total de espaços: 4

Em Python, **há várias funções** que **podemos** usar para ajudar na construção de programas - inclusive para fazer o que o trecho de programa acima faz!

str.lower()

retorna uma **cópia** de str com os caracteres em minúsculas

```
>>> 'CasA'.lower()
'casa'
```

str.upper()

retorna uma **cópia** de str com os caracteres em maiúsculas

```
>>> 'CasA'.upper()
'CASA'
```

str.title()

retorna uma **cópia** de str com os caracteres iniciais das palavras em maiúscula e as demais em minúsculas.

```
>>> frase = 'bom dia! aula DE programação. vamos pRATICAR?'
>>> frase.title()
'Bom Dia! Aula De Programação. Vamos Praticar?'
```

str.isdigit()

True se todos os caracteres forem dígitos

```
>>> '1234567890'.isdigit()
True
>>> '12345-123'.isdigit()
False
```

str.isalpha()

True se todos os caracteres forem alfabéticos

```
>>> 'aBvFdCc'.isalpha()
True
>>> 'ascvf as'.isalpha()
False
```

str.isalnum()

True se todos os caracteres forem dígitos e alfabéticos

```
>>> '123ase3'.isalnum()
True
>>> '123+34'.isalnum()
False
```

str.isspace()

True se existe apenas o caractere espaço

```
>>> ' '.isspace()
True
>>> '   '.isspace()
True
>>> ' . '.isspace()
False
```

Outros métodos:

sintaxe:

`str.count(sub[, start[, end]])`

`str.count()` – contar o número de ocorrências de uma **substring** em **str**

File Edit Format Run Options Window Help

```
''' string '''
```

```
frase = 'Bom dia! Hoje é quarta-feira.'
```

```
print(f'Total de espaços com str.count(): {frase.count(" ")}' )
```

no exemplo a substring é " " representando UM ESPAÇO

```
-- RESIANT, C:/Users/PIOLA/OneDrive/
```

```
Total de espaços com str.count(): 4
```

```
>>>
```

na frase há 4 ocorrências de UM ESPAÇO

str.count() – outros exemplos

File Edit Format Run Options Window Help

```
''' string - uso de str.count() '''
```

```
frase = 'Boa tarde ou boa noite? Hoje é quarta-feira. Uma boa hora para aprender Python'
```

```
print(f'Total de espaços: {frase.count(" ")}' )
```

```
print(f'Total de "boa": {frase.count("boa")}' )
```

```
print(f'Total de "boa/Boa": {frase.lower().count("boa")}' )
```

```
print(f'Total de "boa/Boa até a posição 20": {frase.lower().count("boa",0,20)}' )
```

```
=== RESTART: C:/Users/profa/OneDrive/Area C
```

```
Total de espaços: 13
```

```
Total de "boa": 2
```

```
Total de "boa/Boa": 3
```

```
Total de "boa/Boa até a posição 20": 2
```

```
>>>
```


str.find()

retorna a **posição** de início da **substring** na **string**, se existir e, **-1**, em caso contrário

sintaxe:

str.find(sub[, start[, end]])

File Edit Format Run Options Window Help

''' string - uso de str.find() '''

0

10

20

30

40

50

>>> len(frase)

78

77

frase = 'Boa tarde ou boa noite? Hoje é quarta-feira. Uma boa hora para aprender Python'

print(f'Posição de "ou" na frase: {frase.find("ou")}')

print(f'Posição de "Olá" na frase: {frase.find("Olá")}')

print(f'Posição de "boa/Boa do inicio": {frase.lower().find("boa")}')

print(f'Posição de "boa/Boa" do final: {frase.lower().rfind("boa")}')

print(f'Posição de "boa/Boa" a partir da posição 3: {frase.lower().find("boa",3)}')

str.rfind()

rear

Exercício: programa que descobre todas as posições da **substring** 'boa' na frase.

= RESTART: C:/Users/proia/OneDrive/Area de Traba

Posição de "ou" na frase: 10

Posição de "Olá" na frase: -1

Posição de "boa/Boa do inicio": 0

Posição de "boa/Boa" do final: 49

Posição de "boa/Boa" a partir da posição 3: 13

>>>

vimos anteriormente:

s1 *in* s2 : retorna *True* se s1 está **contida** em s2 e *False* caso contrário

s1 *not in* s2: retorna *True* se s1 **não** estiver **contida** em s2 e *False* caso contrário

diferença:

- operador *in/not in*: usado para saber se existe a **substring** ou não, na **string**;
- método *find()*: usado se deseja saber se **existe e** em qual **posição** está.

Existem vários outros métodos para trabalhar com strings.

Exemplos:

```
str.split(sep=None, maxsplit=-1)
```

retorna uma lista de strings, separadas de acordo com o separador especificado.

```
str.replace(old, new[, count])
```

retorna uma cópia de string com todas as ocorrências da **substring velha** trocada pela **substring nova**

Procure na plataforma de python.org retorna uma lista de strings, separadas de acordo com o separador especificado.

sintaxe:

`str.split(sep=None, maxsplit=-1)`

str.split() retorna uma lista de strings, separadas de acordo com o separador especificado.

File Edit Format Run Options Window Help

''' string - uso de str.split() '''

frase = 'Boa tarde ou boa noite. Hoje é quarta-feira. Uma boa hora para aprender Python'

print(frase.split())

print(frase.split(maxsplit=2))

print(frase.split('.'))

significa: o máximo de vezes que será dividida + o resto da string, se houver – não precisa escrever a palavra *maxsplit* → frase.split(2)

REPLINK: c:/users/profa/onedrive/area de trabalho/Programas Py/string split.py

['Boa', 'tarde', 'ou', 'boa', 'noite.', 'Hoje', 'é', 'quarta-feira.', 'Uma', 'boa', 'hora', 'para', 'aprender', 'Python']

['Boa', 'tarde', 'ou boa noite. Hoje é quarta-feira. Uma boa hora para aprender Python']

['Boa tarde ou boa noite', ' Hoje é quarta-feira', ' Uma boa hora para aprender Python']

>>>

str.*replace*() retorna uma cópia de string com todas as ocorrência da ***substring velha*** trocada pela ***substring nova***

sintaxe:

str.**replace**(old, new[, count])

File Edit Format Run Options Window Help

```
''' string - uso de str.split()'''
```

```
frase = 'Boa tarde ou boa noite. Hoje é quarta-feira. Uma boa hora para aprender Python.'
```

```
frasenova = frase.replace('.', '!')
```

```
print(frase)
```

```
print(frasenova)
```

```
===== RESTART: C:/Users/profa/OneDrive/Area de Trabalho
Boa tarde ou boa noite. Hoje é quarta-feira. Uma boa hora para aprender Python.
Boa tarde ou boa noite! Hoje é quarta-feira! Uma boa hora para aprender Python!
>>>
```