Insper

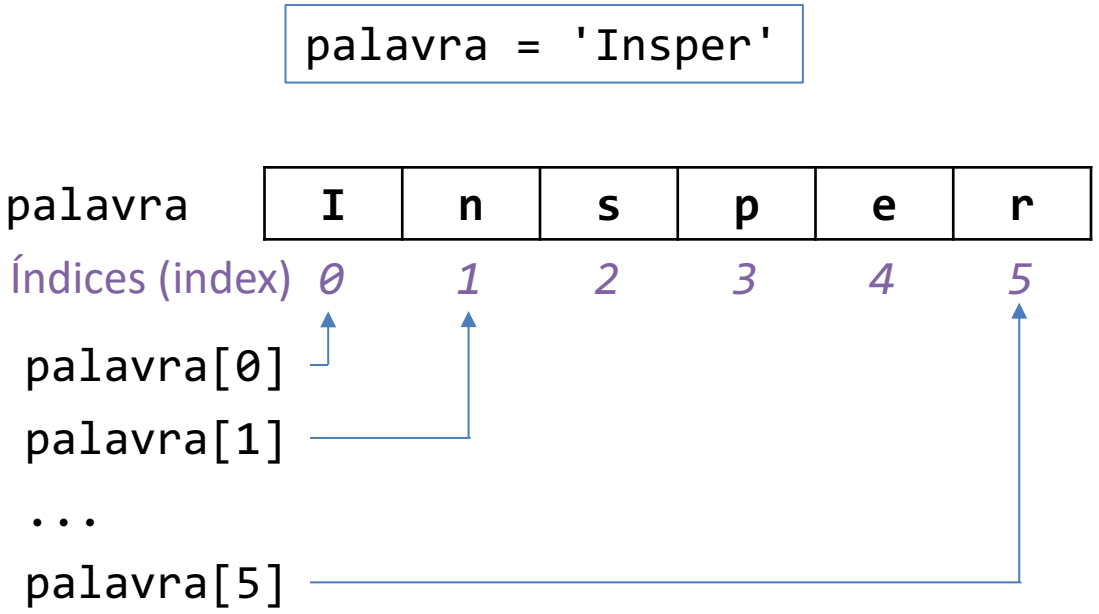
Design de Software

Aula: Strings em Python

2020 – Engenharia

Strings

Cadeias de caracteres



Exemplo

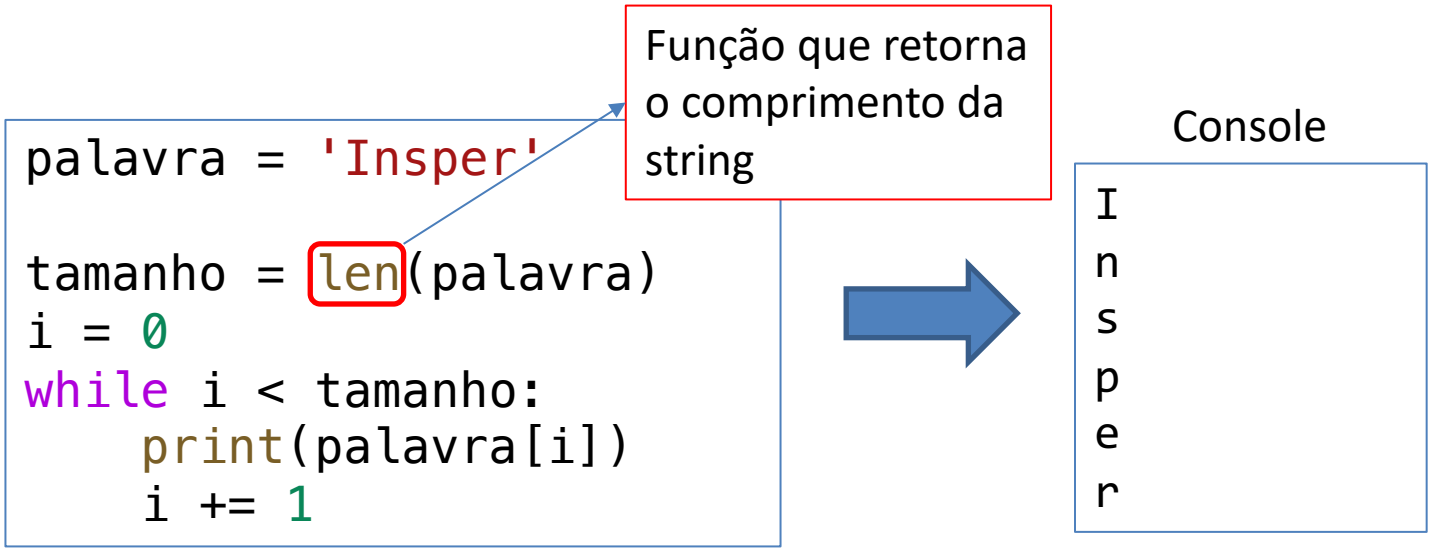
```
palavra = 'Insper'

print(palavra[0])
print(palavra[1])
print(palavra[2])
print(palavra[3])
print(palavra[4])
print(palavra[5])
```

Console

```
I
n
s
p
e
r
```

Exemplo, versão melhorada



Exercício

Faça uma função que recebe uma string e retorna o número de vezes em que a letra ‘a’ aparece nela.

Solução

```
def conta_letra_a(texto):
    contador = 0
    i = 0
    n = len(texto)

    while i < n:
        if texto[i] == 'a':
            contador += 1
        i += 1

    return contador

print(conta_letra_a('abacate'))
print(conta_letra_a('pêssego'))
```

Console

```
3
0
```

find e replace

find: método que retorna a posição da primeira ocorrência de uma dada string em outra. Se não encontrou, retorna -1

replace: método que recebe duas strings e substitui todas as ocorrências da primeira pela segunda

```
s = 'Engenharia Insper'

pos = s.find('Ins')
print(pos)

pos = s.find('abobora')
print(pos)

t = s.replace('Ins', 'Su')
print(t)
```

Console

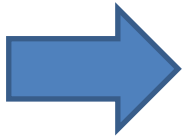
```
11
-1
Engenharia Super
```

strip

Remove caracteres em branco (incluindo o caractere especial '\n') de ambas as pontas da string.

strip não é uma função, mas sim um *método* do *objeto* da *classe* string. (Vamos estudar isso depois quando aprendermos sobre classes.)

```
s = '   Insper     '
print(len(s))
print('|' + s + '|')
print()
t = s.strip()
print(len(t))
print('|' + t + '|')
```



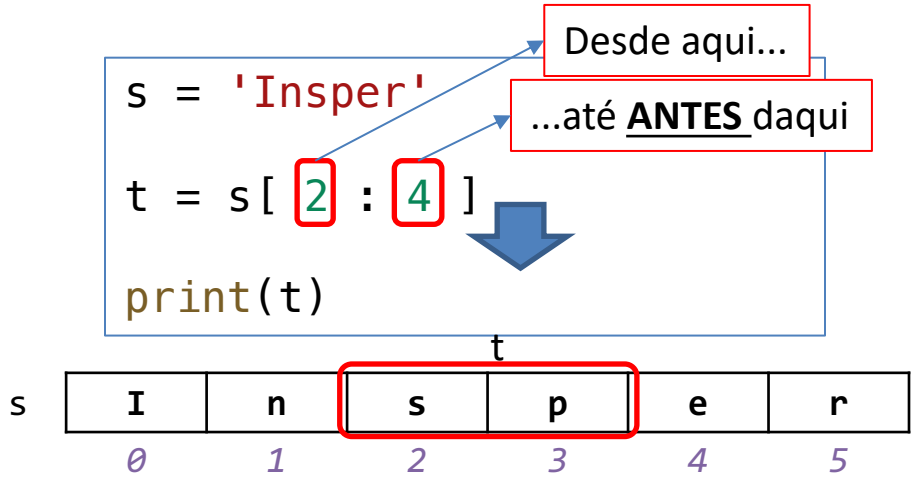
Console

```
14
|   Insper     |

6
|Insper|
```

Operações com Strings

Fatiamento (slicing): extraindo sub-strings



Console

```
sp
```

Mais sobre fatiamento

Primeiro argumento ausente:
‘Desde o início’

Segundo argumento ausente:
‘Até o final’

Ambos argumentos faltantes:
‘Desde o início até o final’

```
s = 'Insper'

t = s[:3]
print(t)

u = s[3:]
print(u)

v = s[:]
print(v)

print(t + u)
print(u + t)
```

memória

s	'Insper'
t	'Ins'
u	'per'
v	'Insper'

Console

```
Ins
per
Insper
Insper
perIns
```

Mais sobre indexação e fatiamento

```
s = 'Engenharia Insper'

print(s[-1])

print(s[ 1 : 17 : 2])

print(s[ : : -1])
```

Dá a volta lá por trás

Pula de 2 em 2

Pula de -1 em -1 (ou seja, anda do fim para o começo)

Console

```
r
nehraIse
repsnI airahnegnE
```

Exercício

Faça uma função que recebe uma string e retorna True se ela for um palíndromo (é a mesma de trás para frente), ou False caso contrário. Por exemplo, a string ‘roma é amor’ é um palíndromo.

Obs: Use fatiamento.

Desafio: dá para fazer essa função com apenas 2 linhas de código.

Solução

```
def verifica_palindromo(texto):
    return texto == texto[::-1]

print(verifica_palindromo('roma é amor'))
print(verifica_palindromo('Insper'))
```

Console

```
True
False
```

Fatiamento de listas

```
dias = ['dom', 'seg', 'ter', 'qua', 'qui', 'sex', 'sab']

dias_uteis = dias[1:6]

fim_de_semana = dias[:1] + dias[6:]

print(dias)
print(dias_uteis)
print(fim_de_semana)
```

Console

```
['dom', 'seg', 'ter', 'qua', 'qui', 'sex', 'sab']
['seg', 'ter', 'qua', 'qui', 'sex']
['dom', 'sab']
```

Exercícios

Faça uma função que recebe uma lista de números reais e retorna uma nova lista contendo apenas os números positivos da lista original

Solução exercício

```
def extrai_positivos(valores):
    valores_pos = []
    i = 0
    n = len(valores)
    while i < n:
        if valores[i] > 0.0:
            valores_pos.append(valores[i])
        i += 1
    return valores_pos

a = [-3.0, 4.2, 7.3, -0.5, 9.0]
b = extrai_positivos(a)
print(a)
print(b)
```

Console

```
[-3.0, 4.2, 7.3, -0.5, 9.0]
[4.2, 7.3, 9.0]
```

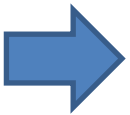
Exercícios

Faça uma função que recebe uma string contendo um e-mail válido e retorna a posição do caractere ‘@’.

Solução exercício

```
def pos_arroba(email):
    pos = -1
    i = 0
    n = len(email)
    while i < n:
        if email[i] == '@':
            pos = i
        i += 1
    return pos

print(pos_arroba('page@google.com'))
```



Console

```
4
```

Exercícios

Faça uma função que recebe uma string contendo um e-mail válido e retorne o nome do usuário.

Use a função do exercício 2. Ou seja, assuma que já existe uma função **`pos_arroba(email)`** nos moldes acima.

Solução exercício

```
def extrai_usuario(email):
    pos = pos_arroba(email)
    return email[:pos]

print(extrai_usuario('page@google.com'))
```

Console

```
page
```

Desafio

Repetir o exercício dos palíndromos sem usar fatiamento

Solução

```
def verifica_palindromo(texto):
    palindromo = True
    i = 0
    j = len(texto) - 1

    while i < j:
        if texto[i] != texto[j]:
            palindromo = False
        i += 1
        j -= 1

    return palindromo

print(verifica_palindromo('roma é amor'))
print(verifica_palindromo('Insper'))
```

Estratégia: vamos começar assumindo que é palindromo, e vamos procurar pares de letras que provem o contrário

O indice **i** começa do inicio e anda para frente

Já o indice **j** começa do final e anda para tras

Enquanto os indices não se encontram...

Se deu diferença de letras, já não é palindromo!

Anda o **i** para frente e o **j** para trás

Quando chega aqui:

- Se alguma letra não bateu entrou no **if** e por isso **palindromo** vai valer **False**
- Se todas as letras bateram, nunca entrou no **if** e então **palindromo** ainda vai estar como **True**

Operações com Strings

Concatenação: “adição” de strings

```
s1 = 'Ins'
s2 = 'per'

s = s1 + s2
print(s)

t = s2 + s1
print(t)
```

Console

```
Insper
perIns
```

Operações com Strings

Repetição: “multiplicação” de strings

```
s1 = 'Ins'
s2 = 'per'

s = 3*s1 + s2
print(s)
```

Console

```
InsInsInsper
```

Exercício

Faça uma função que recebe um inteiro positivo n e retorna uma string contendo uma sequência de n asteriscos.

Solução

```python
def barra_de_asteriscos(n):
    barra = '*' * n
    return barra

print(barra_de_asteriscos(10))
```

Console

```
**********
```

Insper

www.insper.edu.br