



Laboratório de Programação Competitiva

Profa. Silvia Brandão

2024.2

Strings e seus métodos

- `len(string)` - retorna o tamanho de caracteres da string
- `upper()` - retorna um string todo em maiúsculas
- `lower()` - retorna um string todo em minúsculas
- `capitalize()` - retorna um string com o primeiro caractere em maiúscula, e o resto em minúsculas
- `strip()` - retorna um string removendo caracteres em branco do início e do fim
- `lstrip()` - retorna um string removendo caracteres em branco do início
- `rstrip()` - retorna um string removendo caracteres em branco 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

- Você deve experimentar esses métodos para que possa entender o que cada um faz. Observe que os métodos que retornam uma string não alteram a original.
- Dúvidas consulte as referências indicadas.

Exercícios em Python

Considere o código abaixo, em Python:

```
nome = input("Digite seu nome: ")
idade = int(input("Digite sua idade: "))
cidade = input("Digite o nome de sua cidade: ")

apresentacao = f"\nOlá! Meu nome é {nome}. Eu tenho {idade} anos e moro em {cidade}."
print(apresentacao)
```

Determine e imprima:

- quantos caracteres tem a string "apresentacao" para uma dada entrada de dados via teclado;
- troque o nome da cidade para São Paulo, caso seja diferente; e, imprima a string "apresentacao" novamente;
- Troque todas as vogais da string "apresentacao" por '*' e, imprima a mesma novamente.

Aula de hoje ...

- Estruturas de dados em Python: listas, tuplas, dicionários e conjuntos.
- Estruturas de Controle, incluindo Condicionais e Loops.
- Discussão sobre estratégias de resolução de código.



Não se esqueçam do Uniube+

Estruturas de dados em Python:

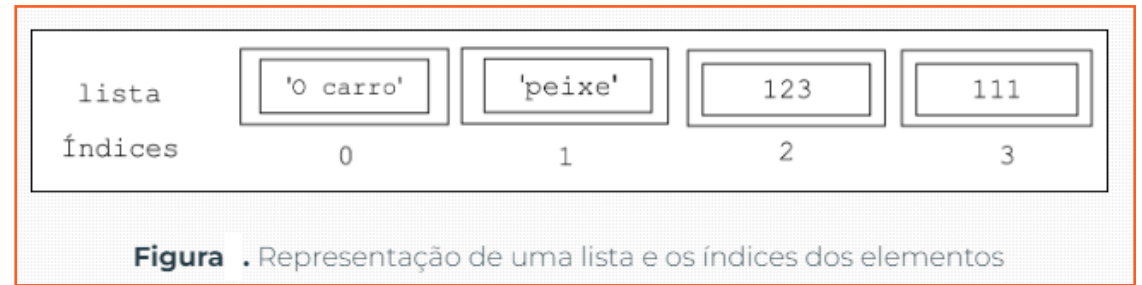
Listas, Tuplas, Conjuntos e Dicionários

- **Listas (list):** são **coleções ordenadas de elementos**, que podem ser de qualquer tipo de dados. Elas **são mutáveis**, o que significa que seus elementos podem ser modificados.
 - Exemplo: [1, 2, 3, 3, 4], ["maçã", "banana", "laranja"], etc.
- **Tuplas (tuple):** são semelhantes às listas, porém **são imutáveis**, ou seja, seus elementos não podem ser alterados após a criação. Elas são representadas por parênteses e podem conter diferentes tipos de dados.
 - Exemplo: (1, 2, 3, 4), ("carro", "moto", "bicicleta"), etc.
- **Conjuntos (set):** representam **coleções não ordenadas e não duplicadas de elementos**. São úteis para realizar operações de união, interseção e diferença entre conjuntos.
 - Exemplo: {1, 2, 3}, {"maçã", "banana", "laranja"}, etc.
- **Dicionários (dict):** são estruturas de dados que **armazenam pares de chave-valor**. Cada valor é associado a uma chave única, permitindo a recuperação eficiente dos dados.
 - Exemplo: {"nome": "João", "idade": 30, "cidade": "São Paulo"}, etc.

Estruturas de dados em Python

Listas e Tuplas

- As listas são mutáveis, o que permite adicionar, remover e modificar elementos. Assim, podemos, por exemplo, alterar o valor de um elemento específico em uma lista, adicionar um novo elemento ao final ou remover um elemento existente.
- Já as tuplas são imutáveis e não permitem tais operações. Veja o exemplo:



```
# Lista (mutável)
lista = [1, 2, 3]
lista[0] = 4
lista.append(5)
lista.remove(2)
print(lista) # Output: [4, 3, 5]

# Tupla (imutável)
tupla = (1, 2, 3)
tupla[0] = 4 # Erro: As tuplas são imutáveis e não podem ser modificadas
```

[4, 3, 5]

```
TypeError                                Traceback (most recent call last)
<ipython-input-7-638788d43a93> in <cell line: 10>()
      8 # Tupla (imutável)
      9 tupla = (1, 2, 3)
----> 10 tupla[0] = 4 # Erro: As tuplas são imutáveis e não podem ser modificadas

TypeError: 'tuple' object does not support item assignment
```

O uso da função range() para gerar sequências de números (list)

DEFINIÇÃO: range(stop[])

DEFINIÇÃO MATEMÁTICA: $[0,5[$

```
1 #gerando uma sequência de 0 à 9
2 list(range(10))
```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```
1 #gerando uma sequência de 1 à 11
2 list(range(1, 11))
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
1 #gerando uma sequência de 0 à 30 com step=5
2 list(range(0, 30, 5))
```

[0, 5, 10, 15, 20, 25]

DEFINIÇÃO: range([start], stop[, step])

DEFINIÇÃO MATEMÁTICA: $[0,5[$

```
1 #gerando uma sequência numérica negativa
2 list(range(0, -10, -1))
```

[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]

```
1 #gerando uma sequência numérica vazia
2 list(range(0))
```

[]

```
1 #gerando uma sequência numérica vazia,
2 #onde o intervalo inicial é maior do que o final
3 #por essa razão, há lista é nula
4 list(range(1, 0))
```

[]

Estruturas de dados em Python

gerando Listas (list) e Tuplas (tuple)

```
▶ squares = []  
  for x in range(10):  
    squares.append(x**2)
```

squares

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

```
▶ tupla = tuple(range(2,10))  
tupla
```

(2, 3, 4, 5, 6, 7, 8, 9)

```
▶ squares = list(map(lambda x: x**2, range(10)))  
squares
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]



O **map** vai aplicar uma função em cada item de uma lista de itens, ou seja, é um for com uma chamada da função para aplicá-la a cada item da sua lista.

Métodos de objetos do tipo lista

- **list.append(x)** - Adiciona um item ao fim da lista. Equivalente a `a[len(a):] = [x]`.
- **list.insert(i, x)** - Insere um item em uma dada posição. O primeiro argumento é o índice do elemento antes do qual será feita a inserção, assim `a.insert(0, x)` insere um elemento na frente da lista e `a.insert(len(a), x)` equivale a `a.append(x)`.
- **list.remove(x)** - Remove o primeiro item encontrado na lista cujo valor é igual a `x`. Se não existir valor igual, uma exceção `ValueError` é levantada.
- **list.pop([i])** - Remove o item na posição fornecida na lista e retorna. Se nenhum índice for especificado, `a.pop()` remove e retorna o último item da lista. Levanta um `IndexError` se a lista estiver vazia ou o índice estiver fora do intervalo da lista.
- **list.clear()** - Remove todos os itens de uma lista. Equivalente a `del a[:]`.
- **list.index(x[, start[, end]])** - Devolve o índice base-zero do primeiro item cujo valor é igual a `x`, levantando `ValueError` se este valor não existe. Os argumentos opcionais `start` e `end` são interpretados como nas notações de fatiamento e são usados para limitar a busca para uma subsequência específica da lista. O índice retornado é calculado relativo ao começo da sequência inteira e não referente ao argumento `start`.

Métodos de objetos do tipo lista

- **list.count(x)** - Devolve o número de vezes em que x aparece na lista.
- **list.sort(*, key=None, reverse=False)** - Ordena os itens na lista (os argumentos podem ser usados para personalizar a ordenação, veja a função sorted() para maiores explicações).
- **list.reverse()** - Inverte a ordem dos elementos na lista.
- **list.copy()** - Devolve uma cópia rasa da lista. Equivalente a a[:].

Exemplos

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4) # Find next banana starting at position 4
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()
'pear'
```

Métodos de objetos do tipo lista

Podemos remover um item de uma lista usando seu índice no lugar do seu valor com a instrução **del**, que difere do método pop() que devolve um valor.

A instrução del pode também ser utilizada para remover fatias de uma lista ou limpar a lista inteira (que fizemos antes por atribuição de uma lista vazia à fatia a[:]). Por exemplo:

```
>>> a = [-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> del a[2:4]
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a
[]
```

del também pode ser usado para remover totalmente uma variável:

```
>>> del a
```

Estruturas Condicionais

if simples

```
a = 33
b = 200
if b > a:
    print("b é maior que a")
```

ifs aninhados


```
x = 41

if x > 10:
    print("maior que 10,")
    if x > 20:
        print("e também maior que 20!")
    else:
        print("mas não maior que 20.")
```

A palavra-chave **elif** é a maneira do Python dizer "se as condições anteriores não eram verdadeiras, tente esta condição".

if composto

```
a = 200
b = 33
if b > a:
    print("b é maior que a")
else:
    print("b não é maior que a")
```



```
1 idade = int(input('Digite sua idade: '))
2 if idade >= 10 and idade < 20:
3     print('Você é adolecente')
4 elif idade >= 20 and idade < 30:
5     print('Você é jovem')
6 elif idade >= 30 and idade <= 100:
7     print('Você é adulto')
8 else:
9     print('Valor não encontrado!')
```

Listagem 1. Estrutura condicional completa com if-elif-else

Estruturas de Repetição - loop

```
[6] 1+2+3+4+5+6+7+8+9+10
```

55

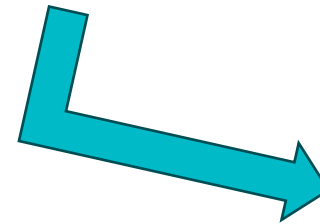


```
soma = 0
numero = 1
while numero <= 10:
    soma = soma + numero
    numero = numero + 1
print(f'Resultado {soma:3d}')
```

Resultado 55

```
nro = 0
while nro <= 10:
    if nro == 5:
        break
    nro+=1
    print('O número é ' + str(nro))

print('Fora do loop')
```



O número é 1
O número é 2
O número é 3
O número é 4
O número é 5
Fora do loop

A instrução **break** é usada para quebrar (ou interromper) o fluxo natural do programa.

Referência

- <https://www.w3schools.com/python/default.asp>
- <https://docs.python.org/pt-br/3/tutorial/index.htm>
- <https://panda.ime.usp.br/pensepy/static/pensepy/>

Faça os exercícios 1015 e 1021 - Beecrowd

```
x1, y1 = map( float, input().split())
```

Estruturas de dados em Python

list - métodos

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4)  # Find next banana starting at position 4
6
```

Exercício:

- Faça um código em Python para a lista fruits acima que :
 - conte a quantidade de strings 'apple';
 - use o método len() para determinar o tamanho da lista;
 - inverta a ordem da lista;
 - remova usando o método pop() o último elemento da lista;
 - por último, ordene a lista.

Acesse o link e estude outros métodos: <https://docs.python.org/pt-br/3/tutorial/datastructures.html>

Exercício:

Gerenciamento de Biblioteca Virtual

Atividade prática: Para o desenvolvimento da biblioteca virtual, vamos optar por utilizar a linguagem de programação Python , com os recursos de listas e tuplas. Por exemplo:

```
livros = ["Dom Casmurro", "O Pequeno Príncipe", "1984", "O Senhor dos Anéis"]
```

```
autores = [("Machado de Assis", 1839), ("Antoine de Saint-Exupéry", 1900), ("George Orwell", 1903), ("J.R.R. Tolkien", 1892)]
```

Assim, é possível adicionar novos livros ao acervo, inserindo títulos na lista livros e informações sobre os autores na tupla autores:

```
livros.append("Harry Potter")  
autores.append(("J.K. Rowling", 1965))
```

- **Crie** uma função de cadastro de livros (no mínimo 5 títulos) em uma lista e armazene os dados sobre os autores desses livros em uma tupla. Em seguida, **crie** uma função de busca que percorra a lista de livros e retorne as informações do livro correspondente, caso este exista.

```
# Exemplo de utilização da função  
titulo_busca = "1984"  
informacoes_livro = buscar_livro(titulo_busca)  
print(f"As informações do livro '{titulo_busca}' são: {informacoes_livro}")
```


Faça os exercícios 1015 e 1021 - Beecrowd

```
x1, y1 = map( float, input().split())
```

Exercícios em Python – Beecrowd 2782

beecrowd | 2782

Escadinha

Por OBI  Brasil

Timelimit: 1



				10	7	4
2	3	4	5			

Dizemos que uma sequência de números é uma escadinha, se a diferença entre números consecutivos é sempre a mesma. Por exemplo, "2, 3, 4, 5" e "10, 7, 4" são escadinhas. Note que qualquer sequência com apenas um ou dois números também é uma escadinha! Neste problema estamos procurando escadinhas em uma sequência maior de números. Dada uma sequência de números, queremos determinar quantas escadinhas existem. Mas só estamos interessados em escadinhas tão longas quanto possível. Por isso, se uma escadinha é um pedaço de outra, consideramos somente a maior. Por exemplo, na sequência "1, 1, 1, 3, 5, 4, 8, 12" temos 4 escadinhas diferentes: "1, 1, 1", "1, 3, 5", "5, 4" e "4, 8, 12".

Entrada

A primeira linha da entrada contém um inteiro **N** ($1 \leq N \leq 1000$) indicando o tamanho da sequência de números. A segunda linha contém **N** inteiros definindo a sequência. O valor dos números da sequência está entre -10^6 e 10^6 inclusive.

Saída

Imprima uma linha contendo um inteiro representando quantas escadinhas existem na sequência.

					4	8	12
				5	4		
		1	3	5			
1	1	1					

Exemplos de Entrada	Exemplos de Saída
8 1 1 1 3 5 4 8 12	4
1 112	1
5 11 -106 -223 -340 -457	1

Leituras para a próxima aula:

- As três principais estruturas de dados utilizadas em ordenação e alocação de dados são as estruturas de dados dinâmicas: lista, fila e pilha. Essas estruturas são fundamentais para o desenvolvimento de aplicações; assim como as árvores.
- <https://docs.python.org/pt-br/3/tutorial/datastructures.html>

Tarefa **MANUSCRITA** para a próxima aula – valor **5pts**

Pesquise e responda as perguntas a seguir, exemplificando em Python:

- a) como se dá a manipulação de uma lista (operações de acesso ao elemento)? Exemplifique.
- b) qual a forma de acesso aos dados de uma fila? E, como acessar um deque? Exemplifique.
- c) qual a forma de acesso aos dados de uma pilha? Exemplifique.
- d) o que é Compreensões de lista? Exemplifique.
- e) vamos explorar os conceitos de dicionários e conjuntos? Explique e exemplifique.

Grupos de 2 alunos.

Entrega do material escrito e apresentação dos tópicos pelos grupos de alunos.