🎯 Aula — Strings em Python

1️⃣ Fatiamento de Strings

Cada **caractere** de uma string tem um índice (posição), começando no **0**.
Exemplo:

```
frase = 'Curso em Video Python'
# índice: 0123456789.....
```

📌 Exemplos de uso:

```
print(frase[9])        # Mostra só o caractere do índice 9 → V
print(frase[9:13])     # Do índice 9 até 12 → 'Vide'
print(frase[9:21])     # Do índice 9 até 20 → 'Video Python'
print(frase[9:21:2])   # Do índice 9 até 20, pulando de 2 em 2 → 'VdoPto'
print(frase[:5])       # Do início até índice 4 → 'Curso'
print(frase[15:])      # Do índice 15 até o final → 'Python'
print(frase[9::3])     # Do índice 9 até o final, de 3 em 3 → 'VePh'
```

2️⃣ Análise de Strings

Funções úteis para **analisar** o texto:

```
print(len(frase))              # Conta o comprimento da string → 21
print(frase.count('o'))        # Conta quantas vezes aparece a letra 'o' → 3
print(frase.count('o', 0, 13))  # Conta 'o' só do índice 0 até 12 → 1
print(frase.find('deo'))       # Mostra o índice do início de 'deo' → 11
print('Curso' in frase)        # Verifica se 'Curso' está dentro da frase →
True
```

3️⃣ Transformação de Strings

Usado para **alterar** o texto:

```
print(frase.replace('Python','Android'))  # Troca uma palavra
print(frase.upper())       # Tudo maiúsculo
print(frase.lower())       # Tudo minúsculo
print(frase.capitalize())  # Só a primeira letra maiúscula
print(frase.title())       # Primeira letra de cada palavra maiúscula
```

👉 **Remoção de espaços:**

```
print(frase.strip())   # Remove espaços do início e do fim
print(frase.rstrip())  # Remove só da direita
print(frase.lstrip())  # Remove só da esquerda
```

👉 **Divisão e junção:**

```
print(frase.split())        # Divide em lista ['Curso','em','Video','Python']
print('-'.join(frase))      # Junta cada letra com um traço
```

```
print(' '.join(frase.split()))  # Junta palavras separadas por espaço
```

4️⃣ Extras que não estavam na sua lista

Além do que você já trouxe, olha mais funções úteis:

```
print(frase.startswith('Curso'))  # True → começa com 'Curso'
print(frase.endswith('Python'))   # True → termina com 'Python'
print(frase.swapcase())           # Inverte maiúsculas e minúsculas
print(frase.isalpha())            # Verifica se é só letras → False (porque tem
espaço)
print(frase.isdigit())            # Verifica se é só números → False
print(frase.isalnum())            # Letras e números sem espaço → False
```

⚡ Desafios Extras

1️⃣ Peça uma frase e mostre:

- Quantos caracteres ela tem sem contar os espaços.
- Quantas vezes aparece a letra **"a"**.
- Em que posição aparece pela primeira vez e pela última vez.

2️⃣ Faça um programa que leia o **nome completo** e mostre:

- Nome com todas as letras maiúsculas.
- Nome com todas minúsculas.
- Quantas letras ao todo (sem contar espaços).
- Quantas letras tem o primeiro nome.

Lista Expandida de Funções/Métodos de Manipulação de Texto em Python

1. **capitalize()**: Torna a primeira letra maiúscula e o resto minúsculo.
2. **casefold()**: Converte para minúsculas, mais agressivo que lower() para comparações em diferentes idiomas.
3. **center()**: Centraliza a string com preenchimento.
4. **count()**: Conta quantas vezes uma substring aparece.
5. **encode()**: Codifica a string em bytes com um encoding específico (ex.: UTF-8).
6. **endswith()**: Verifica se a string termina com um sufixo.
7. **expandtabs()**: Substitui tabulações por espaços, com tamanho configurável.
8. **find()**: Retorna o índice da primeira ocorrência de uma substring (ou -1 se não encontrar).
9. **index()**: Retorna o índice da primeira ocorrência de uma substring (levanta erro se não encontrar).
10. **isalnum()**: Verifica se todos os caracteres são alfanuméricos (letras ou números).
11. **isalpha()**: Verifica se todos os caracteres são letras.
12. **isascii()**: Verifica se todos os caracteres são ASCII.

13. **isdigit()**: Verifica se todos os caracteres são números.
14. **islower()**: Verifica se todos os caracteres alfabéticos estão em minúsculas.
15. **isspace()**: Verifica se a string contém apenas espaços.
16. **istitle()**: Verifica se a string está em formato de título (cada palavra com a primeira letra maiúscula).
17. **isupper()**: Verifica se todos os caracteres alfabéticos estão em maiúsculas.
18. **join()**: Junta uma lista de strings em uma única string.
19. **ljust()**: Alinha a string à esquerda com preenchimento.
20. **lower()**: Converte a string para minúsculas.
21. **lstrip()**: Remove espaços (ou caracteres) do início.
22. **partition()**: Divide a string em três partes: antes, durante e após a primeira ocorrência de um separador.
23. **replace()**: Substitui uma substring por outra.
24. **rfind()**: Retorna o índice da última ocorrência de uma substring (ou -1 se não encontrar).
25. **rindex()**: Retorna o índice da última ocorrência de uma substring (levanta erro se não encontrar).
26. **rjust()**: Alinha a string à direita com preenchimento.
27. **rpartition()**: Divide a string em três partes, a partir da última ocorrência de um separador.
28. **rsplit()**: Divide a string em uma lista, começando da direita, com base em um delimitador.
29. **rstrip()**: Remove espaços (ou caracteres) do fim.
30. **split()**: Divide a string em uma lista com base em um delimitador.
31. **splitlines()**: Divide a string em uma lista, separando por quebras de linha.
32. **startswith()**: Verifica se a string começa com um prefixo.
33. **strip()**: Remove espaços (ou caracteres) do início e fim.
34. **swapcase()**: Inverte maiúsculas para minúsculas e vice-versa.
35. **title()**: Capitaliza a primeira letra de cada palavra.
36. **translate()**: Substitui caracteres com base em uma tabela de mapeamento.
37. **upper()**: Converte a string para maiúsculas.
38. **zfill()**: Preenche com zeros à esquerda até um tamanho específico.

Variações Adicionais com Módulo re (Expressões Regulares)

39. **re.sub()**: Substitui padrões (não apenas substrings fixas) por outro texto.
40. **re.findall()**: Encontra todas as ocorrências de um padrão e retorna uma lista.
41. **re.search()**: Busca a primeira ocorrência de um padrão e retorna um objeto de correspondência.
42. **re.split()**: Divide a string com base em um padrão regex.

Outras Variações de Manipulação

- **f-strings**: Permite formatar strings com valores dinâmicos de forma simples.
- **List comprehensions**: Cria listas manipulando caracteres de uma string (ex.: transformar letras).

- **Métodos combinados**: Usa múltiplos métodos de string em sequência (ex.: strip com replace).
- **Módulo string**: Oferece constantes como string.ascii_letters para manipulação de texto.
- **Módulo textwrap**: Ajusta texto para um tamanho específico, útil para formatação.

Notas

- Os novos métodos adicionados são: casefold(), encode(), expandtabs(), isalnum(), isascii(), islower(), istitle(), isupper(), partition(), rfind(), rindex(), rpartition(), rsplit(), splitlines(), swapcase(), e translate().
- As variações com re e outras técnicas (como f-strings) ampliam as possibilidades para manipulações complexas.
- Se "variações" significar algo mais específico (ex.: mais métodos de bibliotecas, manipulações em outros contextos, ou foco em algo como contar letras), me avise!
- Respondi sua pergunta anterior sobre a letra "A" (quantidade, primeira e última posição) usando count(), find(), e rfind() da lista.