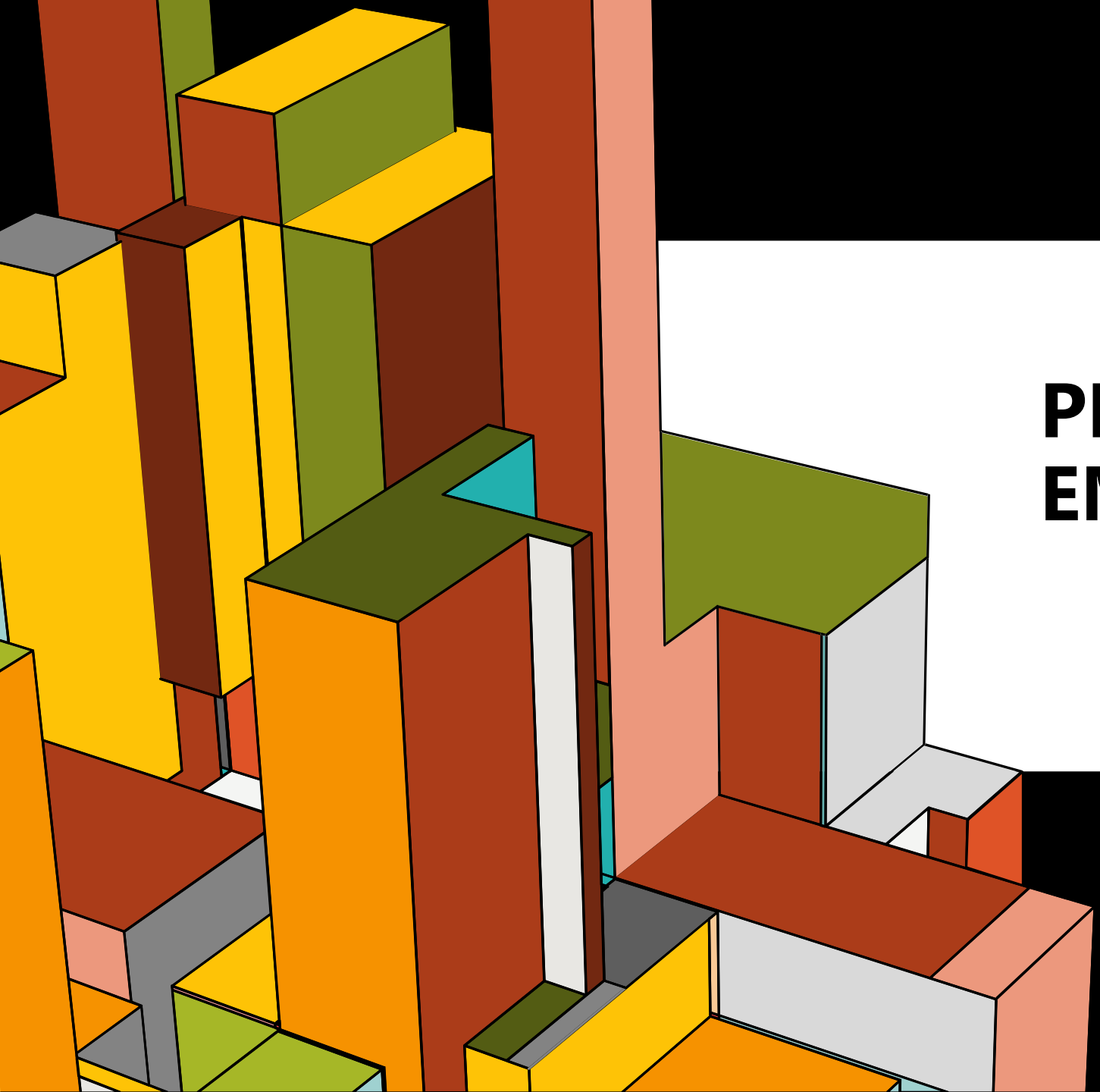

PROGRAMAÇÃO EM PYTHON

python

SENAI

VERIFICAÇÃO DE STRINGS

- **startswith** e **endswith** para validar se a string começa ou termina com uma determinada sequência de caracteres.
- Podemos também verificar se uma palavra está contida na String usando o operador **in**.
- Podemos também verificar se uma palavra não está contida na String usando **not in**

```
cidade = 'São Carlos'
endereco = 'Rua Cândido Padim, 25 - Vila Prado'
completo = cidade + endereco
print(cidade.startswith('São'))
print(cidade.endswith('los'))
print('Rua' in completo)
print('Avenida' not in completo)
```

CONTANDO E PESQUISANDO ELEMENTOS EM UMA STRING

- Podemos contar elementos em uma string usando o método **count**
- Podemos utilizar o método **find** para obter a posição da primeira ocorrência de uma String e podemos utilizar o método **rfind** para realizar a pesquisa da direita para a esquerda
- Tanto **find** quanto **rfind** possuem dois parâmetros para que sejam delimitados o início e o fim para realização da pesquisa
- Para localizar o índice de ocorrência de uma string, podemos usar também o **index** e **rindex**, porém, caso a substring não seja localizada, esses métodos geram uma exceção ValueError

EXEMPLO

```
texto = 'Python é uma linguagem de programação. Python é simples. Python é organizado. Python é uma excelente linguagem.'
print(texto.count('é'))
print(texto.find('Python',25,50))
print(texto.rfind('lingua'))
print(texto.index('é'))
print(texto.rindex('é'))
```

POSICIONAMENTO DE STRINGS

- O método **center** centraliza uma string em um número de posições passado como parâmetro, preenchendo com espaços à direita e à esquerda, até que a string esteja centralizada. Podemos substituir os espaços por qualquer caractere informando como parâmetro para a função.
- O método **ljust** alinha o texto à esquerda e preenche com espaços à direita, já o método **rjust** alinha o texto à direita e preenche com espaços à esquerda

EXEMPLO

```
texto = 'Olá Mundo!'
texto_centro = texto.center(20)
texto_centro_2 = texto.center(20,'=')
texto_esquerda = texto.ljust(12)
texto_direita = texto.rjust(12)
print(f'**{texto_esquerda}*{texto_direita}**')
```

```
**Olá Mundo!    *   Olá Mundo!**
```

12 Espaços

SEPARAÇÃO DE STRINGS

- Para separar strings à partir de um determinado caractere podemos utilizar o método **split**. O resultado é uma lista com os itens que eram delimitados pelo caractere informado na chamada ao método **split**

```
nomes = "João Paulo/Maria Paula/Ana Beatriz/José Pedro"
print(nomes.split('/'))
```

- Para separar strings à partir do caractere de nova linha “\n” podemos usar o método **splitlines**.

```
nomes = "João Paulo\nMaria Paula\nAna Beatriz\nJosé Pedro"
print(nomes.splitlines())
```

SUBSTITUIÇÃO DE STRINGS E REMOÇÃO DE ESPAÇOS EM BRANCO

- Para substituir trechos de uma string podemos usar o método **replace**. Este método recebe por parâmetro a string que será substituída, a nova string, e por fim um número que limitará a quantidade de substituições.
- Se o primeiro parâmetro do **replace** for vazio, será inserido o caractere informado antes de cada caractere da string e se o segundo parâmetro for vazio, o trecho será apagado

EXEMPLO

```
f = 'A força eletromotriz induzida em qualquer circuito fechado é igual ao negativo da variação do fluxo magnético com o tempo na área delimitada pelo circuito'
f1 = f.replace('força', 'Bicicleta')
f2 = f.replace('','#')
print(f1)
print(f2)
```

REMOÇÃO DE ESPAÇOS

```
a = '    Olá mundo    '
print(f'*{a}*')
b = a.strip()
print(f'*{b}*')
c = a.lstrip()
print(f'*{c}*')
d = a.rstrip()
print(f'*{d}*')
```

- Para remover espaços em branco de uma string temos três métodos:
- Strip: Remove espaços no início e fim da string.
- Lstrip: Remove espaços no início da string.
- Rstrip: Remove espaços no fim da string.