re.Search (padrão, string)
↳ procura o padrão em qualquer posição da string
↳ devolve re.match a dizer a posição inicial, final e a **substring** que corresponde ao padrão

re.match (padrão, string)
↳ procura o padrão no começo da string, tipo
↳ devolve re.match a dizer a posição inicial, final e a **substring** que corresponde ao padrão

group (0) → match completo
group (1) → conteúdo do primeiro grupo
group (2) → conteúdo do segundo grupo

exemplo:

```
m = re.search (r'(\d+)-(\d+)-(\d+)', "Data 10-12-2025")
print(m.group(0)) # 10-12-2025
print(m.group(1)) # 10
print(m.group(2)) # 12
print(m.group(3)) # 2025
```

^ → início da string

atribuir nome ao grupo
↳ exemplo
↳ string = 'soy una batata"

```
er7 = re.search(r'(?P<es_oq>batata)', string)
print(er7.group("es_oq")
```

output
↳ batata

↳

```
er8 = re.search(r'(?P<soy_luna>.*>batata', string)
print(er8.group("soy_luna")
```

output
↳ soy una

re.fullmatch → (padrão, string)
↳ retorna match se o padrão corresponder à string toda
↳ exemplo:

```
credit_card_pattern = r'\d{4}-\d{4}-\d{4}-\d{4}'
credit_card_numyer = '1234-5678-9012-3436'
if re.fullmatch (credit_card_pattern, credit_card_number):
    print('Cartão válido!')
else:
    print('Cartão inválido!')
```

↳ output Cartão válido!

re.split(padrão, string)
↳ parte a string de acordo com o padrão envolvendo uma lista de partes
↳ exemplo

re.Findall(padrão, string)
↳ retorna todas as partes que não se sobrepoem e que fazem match com o padrão
↳ exemplo

```
text = "Contact us at support@domain.com or sales@domain.com"
email_pattern = r'\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\b'
email_list = re.findall(email_pattern, text)
print(email_list)
```

↳ output ['support@domain.com', 'sales@domain.com']

re.Finditer(padrão, string)
↳ retorna um iterador sobre as partes que não se sobrepoem e que fazem match com o padrão
↳ exemplo

```
text = "Contact us at support@domain.com or sales@domain.com"
email_pattern = r'\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\b'
email_list = []

for match in re.Finditer(email_pattern, text):
    email_list.append(match)
    start_pos = match.start()
    end_pos = match.end()
    email = match.group()

    print(f"Found email '{email}' at positions {start_pos} - {end_pos - 1}.")

print(email_list)
```

↳ output Found email 'support@domain.com' at positions 14-31.
Found email 'sales@domain.com' at positions 11 31
[<re.match object; span=(14, 32), match='support@domain.com'>, <re.Match object; span=(36, 52), match='sales@domain.com'>]

re.sub(padrão, repl, string)
↳ substitui todas as correspondências de padrão por repl

re.subn(padrão, repl, string)
↳ Faz o msm q sub mas devolve um tuplo(nova_string, num_subs)