Módulo 7 – Strings – Textos e importância no Python – Métodos String – Apresentação (1/9)

```
In [12]: dir(str)

Out[12]: ['__add__',
          '__class__',
          '__contains__',
          '__delattr__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__'
          '__getitem__',
          '__getnewargs__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__mod__',
          '__mul__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__rmod__',
          '__rmul__',
          '__setattr__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'capitalize',
          'casefold',
          'center',
          'count',
          'encode',
          'endswith',
          'expandtabs',
          'find',
          'format',
          'format_map',
          'index',
          'isalnum',
          'isalpha',
          'isascii',
          'isdecimal',
          'isdigit',
          'isidentifier',
          'islower',
          'isnumeric',
          'isprintable',
          'isspace',
          'istitle',
          'isupper',
          'join',
          'ljust',
          'lower',
          'lstrip',
          'maketrans',
          'partition',
          'replace',
          'rfind',
          'rindex',
          'rjust',
          'rpartition',
          'rsplit',
          'rstrip',
          'split',
          'splitlines',
          'startswith',
          'strip',
          'swapcase',
          'title',
          'translate',
          'upper',
          'zfill']
```

Você deve se lembrar que no módulo 5 falamos sobre métodos e o que eles são.

As strings, assim como os outros tipos de variáveis, possuem uma série de métodos que podem ser utilizados para ajudar no tratamento dos dados.

Aqui do lado temos todos os métodos da string e no próximo slide vamos ver os mais importantes.

.CAPITALIZE

```
texto = 'lira'
print(texto.capitalize())
```

```
Lira
```

Transforma APENAS a primeira letra de uma STRING em MAIÚSCULA

Transforma todas as letras MAIÚSCULAS em MINÚSCULAS

.CASEFOLD

```
texto = 'Lira'
print(texto.casefold())
```

```
lira
```

.COUNT

```
texto = 'lira@yahoo.com.br'
print(texto.count('.'))
```

```
2
```

CONTA o número de vezes que um caractere específico aparece na STRING. No caso ao lado '.'

Retorna TRUE (VERDADEIRO) ou FALSE (FALSO) para um teste SE a string termina com uma STRING específica. No exemplo ao lado, como 'lira@**gmail.com** termina com **gmail.com**, o resultado é TRUE.

.ENDSWITH

```
texto = 'lira@gmail.com'
print(texto.endswith('gmail.com'))
```

```
True
```

.FIND

```
texto = 'lira@gmail.com'
print(texto.find('@'))
```

```
4
```

Encontra a posição do termo procurado.

Atenção! Lembre-se que a contagem de posição se inicia em [0]

Já falamos sobre ela anteriormente, lembra?
Ela insere o valor de uma variável no termo indicado por {}. Muito útil para evitar ter que transformar o formato de cada variável individualmente

.FORMAT

```
faturamento = 1000
print('O faturamento da loja foi de {} reais'.format(faturamento))
```

```
O faturamento da loja foi de 1000 reais
```

.ISALNUM

```
texto = 'João123'
print(texto.isalnum())
```

```
True
```

Verifica se um texto é todo feito com caracteres alfanuméricos (letras e números) -> letras com acento ou ç são considerados letras para essa função.

Verifica se um texto é todo feito de letras.
Caso o texto fosse 'João123', o retorno seria FALSE visto que 123 não são letras.

.ISALPHA

```
texto = 'João'
print(texto.isalpha())
```

```
True
```

.ISNUMERIC

```
texto = '123'
print(texto.isnumeric())
```

```
True
```

Verifica se um texto é todo feito por números.

Substitui um caractere escolhido por outro.
No exemplo ao lado, temos que o símbolo PONTO('.') foi alterado por´VÍRGULA(',').
Atenção! Nesse caso, veja que temos 2 argumentos no método. 2 pontos são importantes:

1) A ordem faz diferença;
2) A VÍRGULA indicada em vermelho é o separador dos dois argumentos

.REPLACE

```
texto = '1000.00'
print(texto.replace('.', ','))
```

```
1000,00
```

Separador dos argumentos

.SPLIT

```
texto = 'lira@gmail.com'
print(texto.split('@'))
```

```
['lira', 'gmail.com']
```

Separa o texto da STRING baseado em algum caractere indicado. No caso ao lado, temos a separação do texto antes e depois do '@' .
Perceba que o split já criou uma lista ao fazer essa separação. Isso será bem útil para você

.SPLITLINES

```
texto = '''Olá, bom dia
Venho por meio desse e-mail lhe informar o faturamento da loja no dia de hoje.
Faturamento = R$2.500,00
'''
print(texto.splitlines())
```

```
['Olá, bom dia', 'Venho por meio desse e-mail lhe informar o faturamento da loja no dia de hoje.', 'Faturamento = R$2.500,00']
```

Cria uma lista, onde cada item é o texto de uma linha. Cada "ENTER" é criado um novo item na lista.

.TITLE

```
texto = 'joão paulo lira'
print(texto.title())
```

```
João Paulo Lira
```

Coloca todas as letras iniciais das palavras MAIÚSCULAS

Retira os caracteres indesejados, como por exemplo, espaços que não agregam valor.

Perceba que no resultado fornecido pelo Python não existem os espaços indesejados.

.STRIP

```
texto = ' BEB123453 '
print(texto.strip())
```

```
BEB123453
```

Espaços indesejados

.STARTSWITH

```
texto = 'BEB123453'
print(texto.startswith('BEB'))
```

```
True
```

Retorna TRUE ou FALSE para um teste se uma STRING se inicia com um texto específico.
No caso ao lado, temos que BEB123453 se inicia com BEB, logo, o Python retorna TRUE.

Altera todo o texto para MAIÚSCULAS. Números ficam inalterados.

.UPPER

```
texto = 'beb12343'
print(texto.upper())
```

```
BEB12343
```

Nas aulas anteriores usamos o **format** apenas para substituir as chaves pelos valores das variáveis. Mas o **format** também pode formatar o texto, formatar os números para aparecerem no formato de moeda, porcentagem, com 2 casas decimais, por exemplo.

Vamos ver agora as principais formatações para se usar com o **format.**

Formatações personalizadas com o format

:<	Alinha o texto à esquerda (se tiver espaço na tela para isso)
:>	Alinha o texto à direita (se tiver espaço na tela para isso)
:^	Alinha o texto ao centro (se tiver espaço na tela para isso)
:+	Coloca o sinal sempre na frente do número (independente se é positivo ou negativo)
:,	Coloca a vírgula como separador de milhar
:_	Coloca o _ como separador de milhar
:e	Formato Científico
:f	Número com quantidade fixa de casas decimais
:x	Formato HEX minúscula (para cores)
:X	Formato HEX maiúscula (para cores)
:%	Formato Percentual

- Exemplo de Alinhamento

Tamanho da caixa de texto (em caracteres)

Caixa de texto

```
email = 'lira@gmail.com'
print('Meu e-mail não é {}, show?'.format(email))
```

```
Meu e-mail não é lira@gmail.com, show?
```

Formato padrão **sem alinhamento**

```
email = 'lira@gmail.com'
print('Meu e-mail não é {:<30}, show?'.format(email))
```

```
Meu e-mail não é lira@gmail.com                , show?
```

Caixa de texto com tamanho de **30 caracteres** e texto alinhado **à esquerda (:<)**

```
email = 'lira@gmail.com'
print('Meu e-mail não é {:>30}, show?'.format(email))
```

```
Meu e-mail não é                lira@gmail.com, show?
```

Caixa de texto com tamanho de **30 caracteres** e texto alinhado **à direita (:>)**

```
email = 'lira@gmail.com'
print('Meu e-mail não é {:^30}, show?'.format(email))
```

```
Meu e-mail não é          lira@gmail.com          , show?
```

Caixa de texto com tamanho de **30 caracteres** e texto alinhado **ao centro (:^)**

```
email = 'lira@gmail.com'
print('Meu e-mail não é {:^50}, show?'.format(email))
```

```
Meu e-mail não é                      lira@gmail.com                      , show?
```

Caixa de texto com tamanho de **50 caracteres** e texto alinhado **ao centro (:^)**

- Exemplo de Edição de Sinal

```
custo = 500
faturamento = 270
lucro = faturamento - custo
print('Faturamento foi {:+} e lucro foi {:+}'.format(faturamento, lucro))
```

```
Faturamento foi +270 e lucro foi -230
```

Quando usamos essa formatação (**:+**), sempre será colocado sinal na frente dos números, não importando se esses números são positivos ou negativos.

Nesse caso, o faturamento foi positivo (**+270**), mas o lucro foi negativo (**-230**)

- Exemplo de Separador de Milhar

```
custo = 5000
faturamento = 2700
lucro = faturamento - custo
print('Faturamento foi {:,} e lucro foi {:,}'.format(faturamento, lucro))
```

```
Faturamento foi 2,700 e lucro foi -2,300
```

Quando usamos essa formatação (:,), os números passam a ter separadores de milhar.

Se quiser o sinal e o separador de milhar, devemos fazer como abaixo (:+,)

```
custo = 5000
faturamento = 2700
lucro = faturamento - custo
print('Faturamento foi {:+,} e lucro foi {:+,}'.format(faturamento, lucro))
```

```
Faturamento foi +2,700 e lucro foi -2,300
```

Nesse caso, o faturamento foi positivo (**+2700**), mas o lucro foi negativo (**-2300**)

- Formato com Casas Decimais Fixas

```
custo = 500
faturamento = 270
lucro = faturamento - custo
print('Faturamento foi {:f} e lucro foi {:f}'.format(faturamento, lucro))
```

```
Faturamento foi 270.000000 e lucro foi -230.000000
```

Por padrão, quando rodamos o código com (**:f**), os números passam a ter 6 casas decimais. Mas se quisermos definir a quantidade de casas decimais, devemos usar a seguinte formatação:

:.[número de casas decimais]f

Por exemplo, (**:.2f**) para 2 casas decimais, (**:.1f**) para 1 casa decimal. É **muito importante** não esquecer do **ponto** (**.**)

```
custo = 500
faturamento = 270
lucro = faturamento - custo
print('Faturamento foi {:.2f} e lucro foi {:.1f}'.format(faturamento, lucro))
```

```
Faturamento foi 270.00 e lucro foi -230.0
```

- Formato Percentual

```
custo = 500
faturamento = 1300
lucro = faturamento - custo
margem = lucro / faturamento
print('Margem de lucro foi de {:%}'.format(margem))
```

```
Margem de lucro foi de 61.538462%
```

Quando usamos essa formatação (:%), os números passam a ter o formato percentual

Para ajustarmos também o número de casas decimais, devemos combinar essas duas formatações como na imagem abaixo.

```
custo = 500
faturamento = 1300
lucro = faturamento - custo
margem = lucro / faturamento
print('Margem de lucro foi de {:.2%}'.format(margem))
```

```
Margem de lucro foi de 61.54%
```

- Formato Moeda -> Combinação de Formatos

```
custo = 5000
faturamento = 27000
lucro = faturamento - custo
print('Faturamento foi R$ {:,.2f} e lucro foi R$ {:,.2f}'.format(faturamento, lucro))
```

```
Faturamento foi R$ 27,000.00 e lucro foi R$ 22,000.00
```

Com a formatação usada (**:,.2f**), os números passam a ter o formato de milhar e 2 casas decimais. Mas e se quisermos colocar esses valores no formato brasileiro (**ponto** como separador de milhar e **vírgula** para separar casa decimal)?

Existem módulos que fazem isso, mas podemos usar o método de string **.replace()**.
Vamos substituir a vírgula por ponto, e o ponto por vírgula. Para isso, em vez de usarmos o separador de milhar **vírgula, devemos usar o** separador de milhar **underline**. Isso permitirá a substituição com o **replace**, caso contrário ocorreria um conflito, pois não é possível fazer 2 substituições simultâneas (substituir vírgula por ponto e ponto por vírgula, ao mesmo tempo).

```
#transformando no formato brasileiro
lucro_texto = 'R${:_.2f}'.format(lucro)
print(lucro_texto.replace('.', ',').replace('_', '.'))
```

```
R$22.000,00
```

Lembrando que lucro_texto é do tipo string

- Função **round()** para arredondar números, caso seja necessário

Quando definimos o número de casas decimais usando a formatação (por exemplo, **:.2f**) dentro do format, o valor é automaticamente arredondado. Mas existe também a função **round**, que arredonda um valor de acordo com o número de casas decimais.

Valor que será arredondado

Número de casas decimais para o qual o valor deve ser arredondado

```
imposto = 0.15758
preco = 100
valor_imposto = round(preco * imposto, 1)
print('Imposto sobre o preço é de {}'.format(valor_imposto))
```

```
Imposto sobre o preço é de 15.8
```