F-Strings

Prof. Alberto Costa Neto

Programação em Python

F-Strings

- Oferecem formas de manipular e formatar Strings.
- Surgiram na versão 3.6 de Python.
- Provê uma sintaxe mais legível, concisa e rápida em comparação com métodos mais antigos como o operador % e o método str.format()

Sintaxe Básica

- Para criar uma f-string, basta colocar a letra f ou F como prefixo de uma string
- Dentro da string podemos colocar expressões Python entre chaves { }, cujos valores serão calculados e substituídos na string. Por exemplo:

```
>>> nome = input('Digite o seu nome:')
Jose
>>> idade = int(input('Digite sua idade:'))
34
>>> print(f'{nome} está com {idade} anos!')
Jose está com 34 anos!
```

Expressões

- É possível colocar qualquer expressão que retorne um valor entre chaves.
- A expressão será avaliada e o valor resultante será incluído na string.
- Por exemplo:

```
>>> valor = float(input('Digite o valor do item:'))
10.8
>>> quant = int(input('Digite a quantidade de itens:'))
5
>>> print(f'O total da conta é R$ {valor * quant}!')
O total da conta é R$ 54.0!
```

Formatação de Reais

- A formatação segue basicamente as mesmas regras que a do operador %. Alguns exemplos:

```
>>> pi = 3.14159265
>>> print(f'{pi:10.2f}') # 10 caracteres com 3 casas decimais
      3.14
>>> print(f'{pi:.3f}') # 3 casas decimais
3.142

>>> num_grande = float(input())
123456789.01234
>>> print(f'{num_grande:,.2f}') # milhar e 2 casas decimais
123,456,789.01
```

Formatação de Inteiros

- Permite separar os milhares usando vírgula ou sublinhado_

```
>>> num_grande = 123456789

>>> print(f'{num_grande:,}') # milhar
123,456,789

>>> print(f'{num_grande:_}') # _ no lugar da vírgula de milhar
123_456_789
```

Alinhamento e Preenchimento

- As F-Strings permitem alinhar um texto dentro de uma string
- É possível alinhar à esquerda, à direita e centralizado
- Também é possível definir o caractere a ser usado no preenchimento dos espaços vazios

Alinhamento e Preenchimento

```python
texto = "Python"
print(f"|{texto:<12}|") # Alinhado à esquerda em 12 caracteres
print(f"|{texto:^12}|") # Centralizado em 12 caracteres
print(f"|{texto:->12}|") # Alinhado à direita com preenchimento de '-'
print(f"|{texto:*^12}|") # Centralizado e com preenchimento de '*'
```

```
|Python      |
|   Python   |
|------Python|
|***Python***|
```

Sintaxe:<string>:<carac-preencher><direcao><tam>

- <carac-preencher>: caractere que será usado para completar o espaço definido em <tam> que excede a <string>. O padrão é espaço..
- <direcao>: Pode ser uma das 3 opções < (esquerda), > (direita) ou ^ (centralizado)
- <tam>: indica a quantidade de caracteres totais

F-Strings

- Sugerimos a utilização de F-Strings por estar sendo bastante utilizada e pela sintaxe mais legível, concisa e rápida
- É importante considerar que só está disponível a partir da versão 3.6 de Python.