PYTHON ACADEMY

MANIPULAÇÃO DE STRINGS COM F-STRINGS NO PYTHON

Aprenda a manipular strings em Python de um jeito fácil e descomplicado com f-strings!

PYTHONACADEMY.COM.BR

Olá Pythonista!

Hoje vamos aprender uma forma de manipular strings de um jeito legível e mais conciso, utilizando **f-strings**!

F-strings foram introduzidas no Python 3.6 através da PEP 498.

Ao fim desse post, você aprenderá como e porquê utilizar *f-strings*!

Então faz o cafézinho nosso de cada dia e vamos nessa!

Formatação de Strings em Python

Em Python nós não temos muitas formas de formatar strings, graças a um dos Zen's do Python (não sabe qual? Então já clica aqui pra saber mais).

Antes do Python 3.6, nós tínhamos basicamente duas formas de formatar strings: - Utilizando `%` **ou** - Utilizando `str.format()`, a partir do Python 3.0.

A partir da versão 3.6 do Python, foi introduzido o conceito de *f-strings*, que veremos **AGORA**!

Formatação com *f-strings*

F-strings foram criados para facilitar nossa vida e vieram para **ficar**!

Também chamadas de "strings literais formatadas" (*formatted string literals*), f-strings são strings com a letra `f` no início e chaves `{}` para realizar a interpolação de expressões.

As expressões são processadas em tempo de execução e formatadas utilizadas o protocolo `__format__`. Vamos de exemplo:

```
nome = 'Python Academy'

print(f"Qual o melhor Blog sobre Python? {nome}!")
```

E a saída seria:

```
Qual o melhor Blog sobre Python? Python Academy!!!
```

Utilizando funções

Como *f-strings* são processadas em tempo de execução, podemos colocar ~~quase~~ todo tipo de código dentro das expressões.

Aqui um outro exemplo, utilizando chamada de função e mais:

```
nome = 'python academy'

print(f"Qual o melhor Blog sobre Python? {nome.upper() + '!' * 3}")
```

Sua saída seria:

```
Qual o melhor Blog sobre Python? PYTHON ACADEMY!!!
```

Ou ainda:

```
import math

x = 0.5

print(f'cos({x}) = {math.cos(x)}')
```

O *output* seria:

```
cos(0.5) = 0.8775825618903728
```

Acessando dicionários

Também é possível acessar dicionários dentro de *f-strings*:

```
dicionario = dict({'nome': 'Vinícius', 'ocupacao': 'Software Engi-
        neer'})

print(f"{dicionario['nome']} é um {dicionario['ocupacao']}")
```

Seu output seria:

```
Vinícius é um Software Engineer
```

Strings multi-linha

Também podemos criar *f-strings* multilinha:

```
site = 'Python Academy'
titulo = 'f-string no Python'
dificuldade = 'Básico'

print(
  f"Site: {site}\n"
  f"Título: {titulo}\n"
  f"Dificuldade: {dificuldade}"
)
```

A saída seria a seguinte:

```
Site: Python Academy
Título: f-string no Python
Dificuldade: Básico
```

Vamos nessa! 😉

Método de classe `__str__` vs `__repr__`

Você pode até mesmo utilizar objetos instanciados dentro de *f-strings*. Por exemplo, caso você tenha a seguinte classe:

```
class Carro:
  def __init__(self, marca, modelo, ano):
    self.marca = marca
    self.modelo = modelo
    self.ano = ano

  def __str__(self):
    return f"{self.marca}/{self.modelo} - Ano {self.ano}"

  def __repr__(self):
    return (
      f"Marca: {self.marca}\n"
      f"Modelo: {self.modelo}\n"
      f"Ano: {self.ano}"
    )
```

Seria possível fazer:

```
possante = Carro('Ferrari', 'F8 Tributo', '21')

print(f'{possante}')
```

A saída de código seria:

```
Ferrari/F8 Tributo - Ano 21
```

A saída padrão é a do método `__str__`.

Contudo, se quisermos apresentar a representação presente no método `__repr__`, podemos utilizar *flag* especial `!r`.

Veja como:

```
print(f'{possante!r}')
```

Dessa forma, a saída seria a seguinte:

```
Marca: Ferrari
Modelo: F8 Tributo
Ano: 21
```

> 💡 *Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Depois de ler, dá uma passada no site!*

Utilizando formatadores especiais

A *Especificação de Formatação* (do inglês "*Format Specification*" - acesse a documentação aqui) oferece modificadores que podem ser utilizados em conjunto com *f-strings*.

A especificação é bem extensa e contém diversos componentes, portanto sugiro dar uma olhadinha lá.

Sua forma é a seguinte:

```
{[<nome>][!<conversão>][:<modificador>]}
```

A parte `[:<modificador>]` é bem complexa e possui os seguintes campos:

```
:[[<preenchimento>]<alinhamento>][<sinal>][#][0][<comprimento>][<gru-
po>][.<precisão>][<tipo>]
```

Cada campo desse possibilita um tipo de modificação na string resultante.

Vamos de exemplo!

Um modificador disponível é o símbolo de porcentagem `%`. Ele serve para formatar saídas numéricas. Veja a mágica:

```
valor = 5.5 / 40.0

print(
  f'Resultado original: {valor}\n'
  f'Resultado formatado: {valor:.1%}'
)
```

Olha a saída:

```
Resultado original: 0.1375
Resultado formatado: 13.8%
```

Explicando:

- O `.1` diz que a string resultante deve ter apenas uma casa decimal;
- O `%` multiplica o valor por 100 e inclui o **%** ao final.

Agora um exemplo maluco:

```
valor = 255

print(f"'{valor:-^10x}'")
```

E a saída:

```
'----ff----'
```

Agora vamos com calma:

- `-` é o `[<preenchimento>]`: é esse caracter que vai preencher os espaços vazios;
- `^` é o `[<alinhamento>]`: diz como a string deve ser alinhada. No caso, `^` diz que a string deve ser centralizada.
- `10` é o `[<comprimento>]`: diz que a string resultante deve ter 10 caracteres.
- `x` é o `[<tipo>]`: diz que a string deve ser convertida em hexadecimal (portanto `ff` no resultado).

Um tanto complexo, mas conciso!

Conclusão

Vimos nesse post como é simples utilizar *f-strings* e como deixa nosso código mais legível!

Agora que você sabe como é simples utilizar *f-strings*, que tal refatorar aquele monte de string formatada com `%`?

Uma boa né?! 😅

E aí, que ver mais do quê por aqui? **Comenta aqui embaixo**! 👇

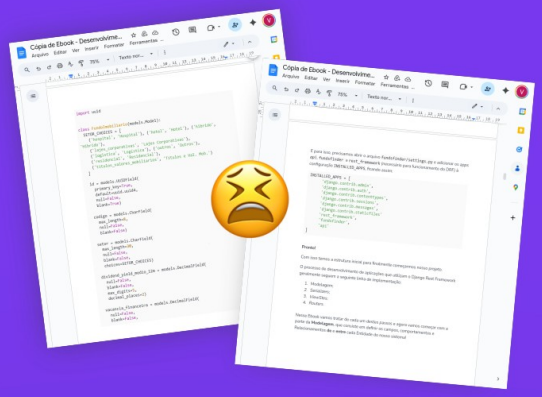
Até a próxima, **Pythonista**!

Não se esqueça de conferir!

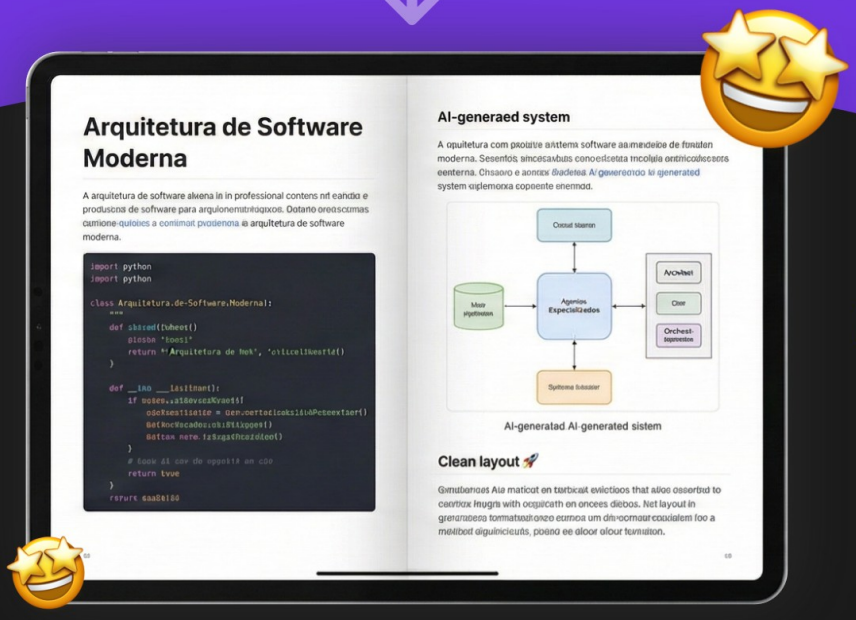
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS