Interpolação e Formatação de textos

Uma das coisas mais úteis de se fazer com texto é a interpolação de variáveis dentro do texto e a formatação de acordo com template pre-definido.

Interpolação é uma alternativa a **concatenação**, enquanto a concatenação usa o sinal de `+` como em `"Bruno" + "Rocha"` na interpolação usamos templates com posicionamento.

Python oferece 3 maneiras de fazer **Interpolação**, a primeira e mais antiga delas segue um padrão universal adotado em muitos sistemas e em outras linguagens de programação e utiliza o sinal de `%` como marcador de template.

%

```
>>> mensagem = "Olá %s, você é o participante número %d e pode ganhar %f pontos."
>>> nome = "Bruno"
>>> numero = 4
>>> pontos = 42.5
>>> print(mensagem % (nome, numero, pontos))
Olá Bruno, você é o participante número 4 e pode ganhar 42.500000 pontos.
```

Este tipo de formatação usa o `%` acompanhado de `s` para str, `d` para int, ou `f` para float, e além de demarcar o **placeholder** onde a substituição irá ocorrer também podemos definir a precisão numérica como em `%.2f` que significa que queremos imprimir apenas 2 casas decimais do número float.

```
>>> mensagem = "Olá %s, você é o participante número %d e pode ganhar %.2f pontos."
>>> print(mensagem % (nome, numero, pontos))
Olá Bruno, você é o participante número 4 e pode ganhar 42.50 pontos.
```

E também é possivel utilizar parâmetros nomeados.

```
>>> mensagem = "Olá %(nome)s, você é o participante número %(num)d e pode ganhar %(pon).2f pon
tos."
>>> print(mensagem % {"nome": "Bruno", "num": 4, "pon": 42.5})
Olá Bruno, você é o participante número 4 e pode ganhar 42.50 pontos.
```

Apesar do uso de `%` ter caido em desuso no Python3, ainda existem bibliotecas como a `logging` que ainda utiliza este formato.

format
Esta é a forma preferida para fazer interpolação de textos pois além de permitir a substituição de variáveis também permite a formatação dos valores, vejamos alguns exemplos:

```
>>> mensagem = "Olá {}, você é o participante número {} e pode ganhar {} pontos."
>>> print(mensagem.format(nome, numero, pontos))
Olá Bruno, você é o participante número 4 e pode ganhar 42.5 pontos.
```

Repare que ao invés de `%` agora usamos `{}` para marcar um **placeholder** e ao inves de `%` usamos a chamada do método `.format` do próprio tipo `str` para passar os valores em sequência.

E também podemos especificar tipos e a precisão numérica usando `:` e os mesmos marcadores dentro de `{}`.

```
>>> mensagem = "Olá {:s}, você é o participante número {:d} e pode ganhar {:.2f} pontos."
>>> print(mensagem.format(nome, numero, pontos))
Olá Bruno, você é o participante número 4 e pode ganhar 42.50 pontos.
```

Podemos utilizar outras opções de formatação em cada uma das marcações entre `{}`. existe toda uma mini linguagem de formatação:

```
{[[fill]align][sign][#][0][width][grouping_option][.precision][type]}
fill - <any character>
align - "<" | ">" | "=" | "^"
sign - "+" | "-" | " "
width - digit+
grouping_option - "_" | ","
precision - digit+
type - "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"
```

Exemplos:

```
# Centralizar fazendo ocupar exatamente 11 caracteres.
>>> "{:^11}".format("Bruno")
'   Bruno   '

# A mesma coisa porém alinhado à direita.
>>> "{:>11}".format("Bruno")
'      Bruno'

# Agora preenchendo os espaços com outro carectere
>>> "{:*^11}".format("Bruno")
'***Bruno***'

# Definindo tipo e precisão para números
>>> "{:*^11.2f}".format(45.30000041)
'***45.30***'
```

O site Pyformat https://pyformat.info/ oferece um guia bastante intuitivo para utilizar as opções de formatação, elas são tantas que não daria para abordarmos todas elas neste treinamento, mas não se preocupe que durante os nossos exercícios vamos utilizar as mais comuns.

Uma outra forma mais rápida de obter essa ajuda é abrindo o python e digitando

```
help('FORMATTING')
```

f strings

No Python 3 foi introduzido um atalho bastante útil para usar o `format` e de uma forma mais natural agora podemos escrever strings que se auto formatam usando as variáveis existentes, o funcionamento respeita as mesmas opções vistas anteriormente, o que muda é só a forma de escrever, ao invés de chamar explicitamente `.format()` usamos `f"texto"`.

```
# Texto
>>> nome = "Bruno"
>>> f"{nome:*^11}"
```

```
'***Bruno***

# Número
>>> valor = 45.30000041
>>> f"{valor:*^11.2f}"
'***45.30***'
```

Uma útilidade interessante das f-strings é usar para fazer debugging.

```
>>> nome = "Bruno"
>>> print(f"{nome=}")
nome='Bruno'
```

Durante o treinamento usaremos:

- `%s` para logs
- `.format` para templates salvos, por exemplo para enviar e-mails
- `f-string` para todas as outras mensagens do programa

Emojis

print("\U0001F600") print("\U0001F43C") print("\N{watermelon}")