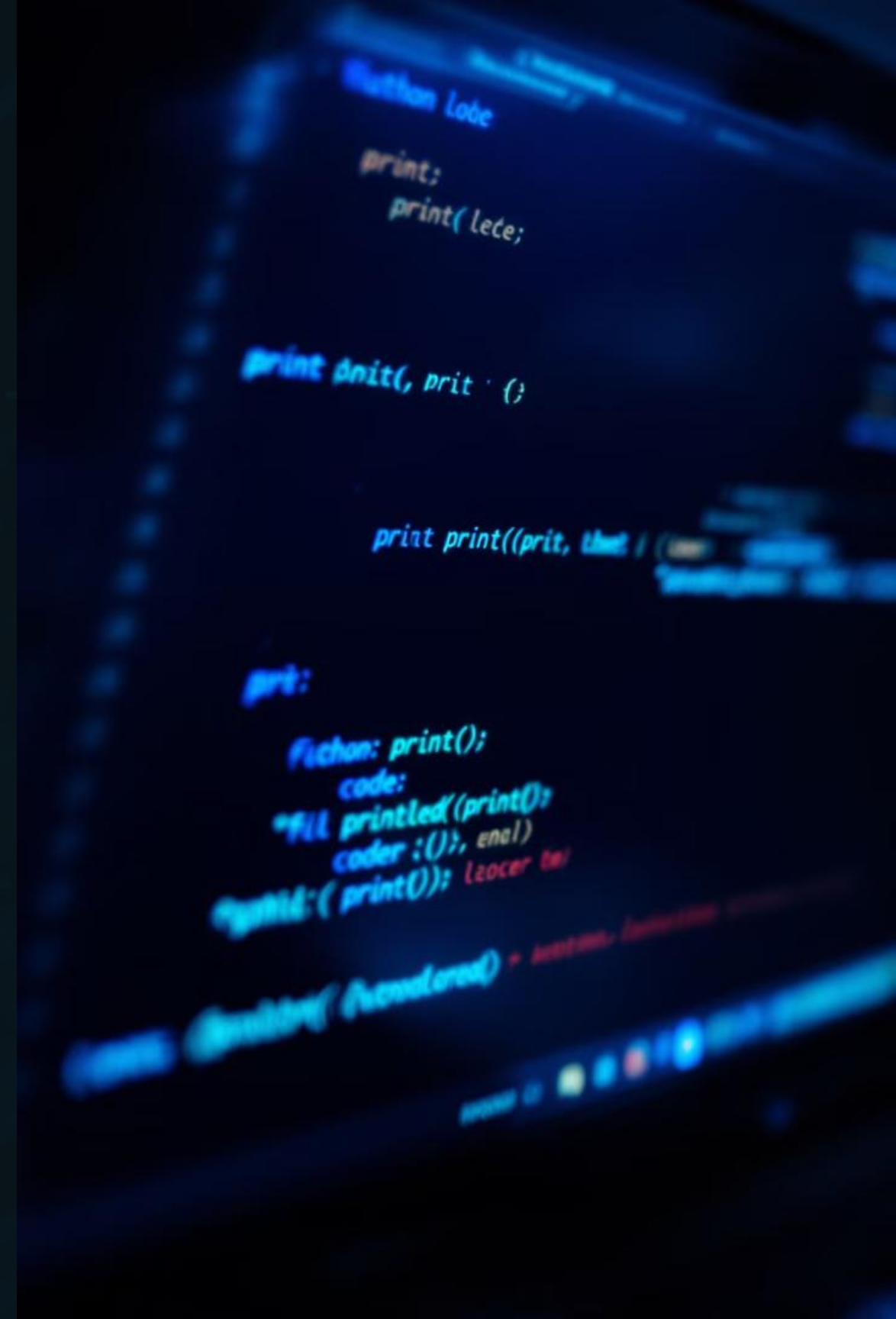


Desvendando a Função `print()` em Python

A função `print()` é uma ferramenta fundamental em Python, atuando como uma "voz" do seu código. Ela permite exibir informações na tela, o que é crucial para entender o que o programa está fazendo. Este guia detalhado explora a fundo essa função essencial, desde sua sintaxe básica até aplicações avançadas, capacitando você a utilizá-la com maestria em seus projetos.

 por **ROBERTO FABIANO FERNANDES**



Sintaxe e Parâmetros da Função print()

`*objects`

Os valores (strings, números) a serem impressos. Aceita um ou vários valores.

`sep`

Define o separador entre os objetos (padrão: ' '). Ex: `sep="-"`.

`end`

Define o que é adicionado ao final (padrão: '\\n' - nova linha). Ex: `end=" "`.

`file`

Especifica onde a saída é enviada (padrão: `sys.stdout` - tela). Pode ser um arquivo.

```
print ( Hello , world!!)
```

```
print (f"Hello {name}");
```

```
for line in sys.stdin:
```

```
    line = line.strip()
```

Exemplo 1: Impressão Simples

Código

```
print("Olá, mundo!")
```

Resultado

```
Olá, mundo!
```

Explicação

Imprime a string "Olá, mundo!" na tela. É o ponto de partida para exibir informações básicas.

No, load the `usep` parameter of in the `hep` the `print` decorator in separation that inserts in the function, multiplication multipliers between multiple objects and being `print`.

```

wil pittws 1. =(61, = 1:ont(3,f(12);

prntlc a: multtewesn, molltwiias = pr1(1),)

conterm: a (int 2 = 2np;

conter as a lint multtcrare, chrt, ind the (ndirn(s),

conterm: multtewen = printa(; × a (1,62);,

conter us a (into trint of, al px.(1);

conter u: multtewenn, printin, = 1 multive(print),

wilttirs: a (inv61 = 61}

prnter is muctrowen + print,}

```

seep lolitive

```

buinttriand + print cor up

b"1 in " is => a print(ant(tint), '
  a pritt) + "in(ta(ple( 1, ad));
}

```

```
printly a noval waten = sep;
wiril = print, + sep
printnt = 21;
```

Exemplo 2: Múltiplos Objetos e o Parâmetro sep

1

Código

```
print("Python", "é", "divertido!")
```

2

Resultado

Python é divertido!

3

Código com sep

```
print("Python", "é", "divertido!", sep="-")
```

4

Resultado com sep

Python-é-divertido!

Exemplo 3: Modificando o Final da Linha com end

print()

```
/"Hello world/">

/== "Hello world("/n
-- "Hello world("/n ;
"= fildl);

(antizille_wrrld) {
ien in aafter:
senn ("an
-- Hello world);
an" spacee_world);;
"Hello world after;;

(inen after < (')
space
warilld, = wafter

"I_space in => wifter
aut:
- printt";
"Hello world);;
- (sfttr_arrrl);

(onsasitions }
and creatix:
/
"a end:
space {
space
in
}
```

print()

```
/" Hello world)>

/== "Hello world))")
-- "Hello wor("n
"= ffilld);

(intizille,,wifir {
ion an spaice :
senn ("in
-- "printe world;
"an" Hello_world);
"Hello world arftr";

(inen after < (')
space
warillo, = wafter

"I_space for "waker"
aft:
- print;
printlo_world);;
in - print);

consaations {
space affer;;
/
"a'en:
space {
after
un
}
```

Código

```
print("Olá", end=" ")
print("mundo!")
```

Resultado

Olá mundo!

Explicação

O parâmetro `end=" "` impede que o `print()` pule para a próxima linha.

Exemplo 4: Redirecionando a Saída com file

Código

```
with open("saida.txt",  
        "w") as arquivo:  
    print("Mensagem no  
arquivo.", file=arquivo)
```



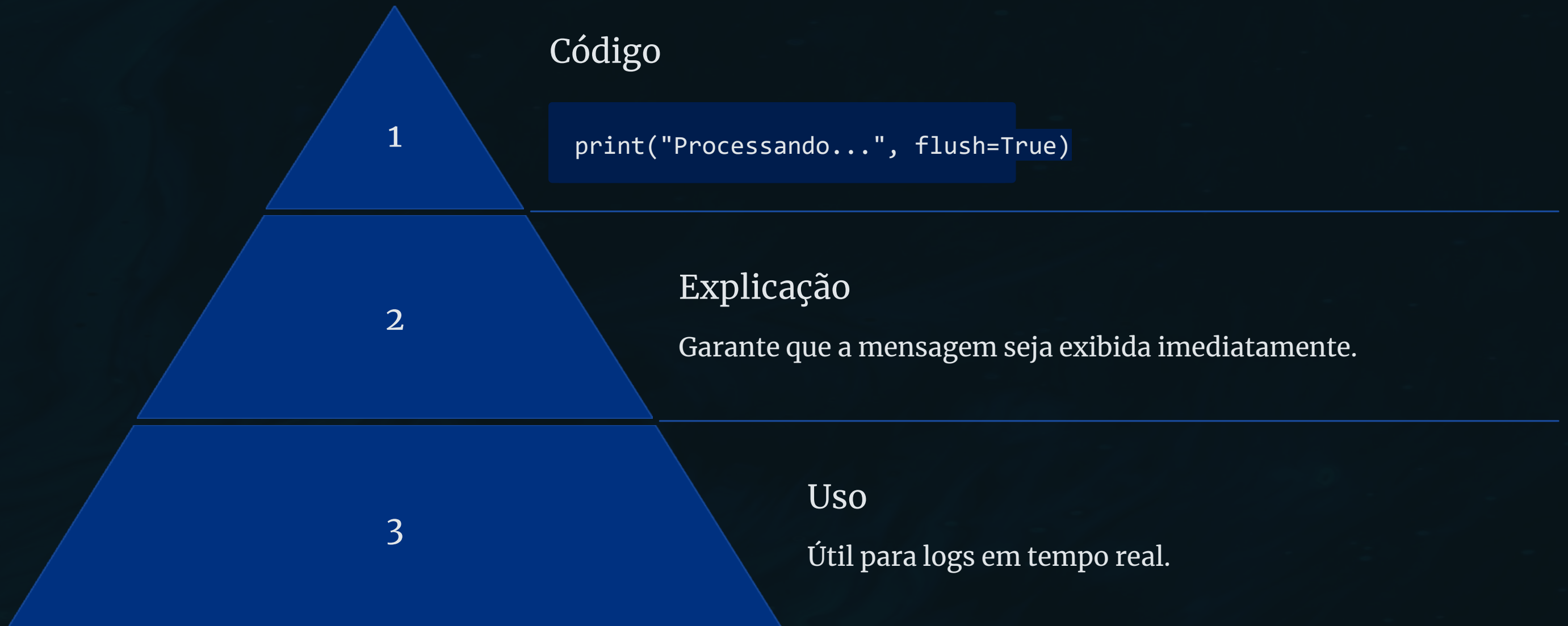
Resultado

Um arquivo chamado **saida.txt** é criado com a mensagem.

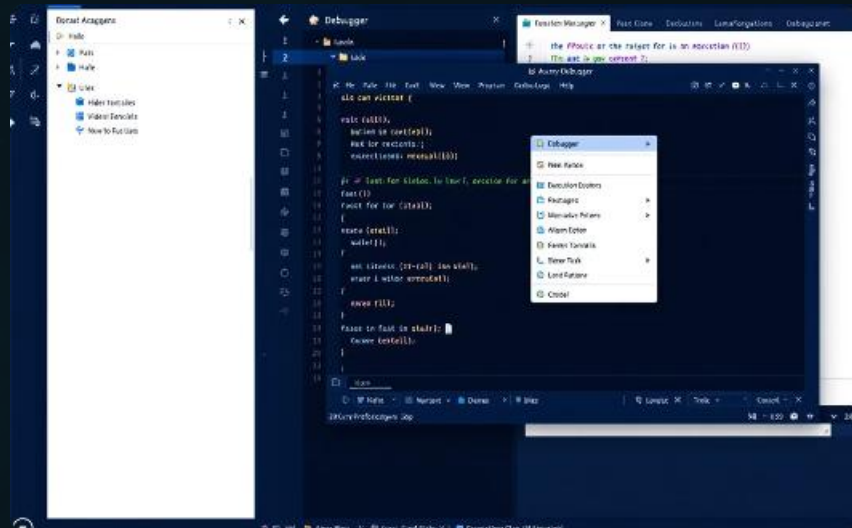
Explicação

A saída é redirecionada para um arquivo, em vez de ser exibida na tela.

Exemplo 5: Forçando a Impressão Imediata com flush



Aplicações da Função print()



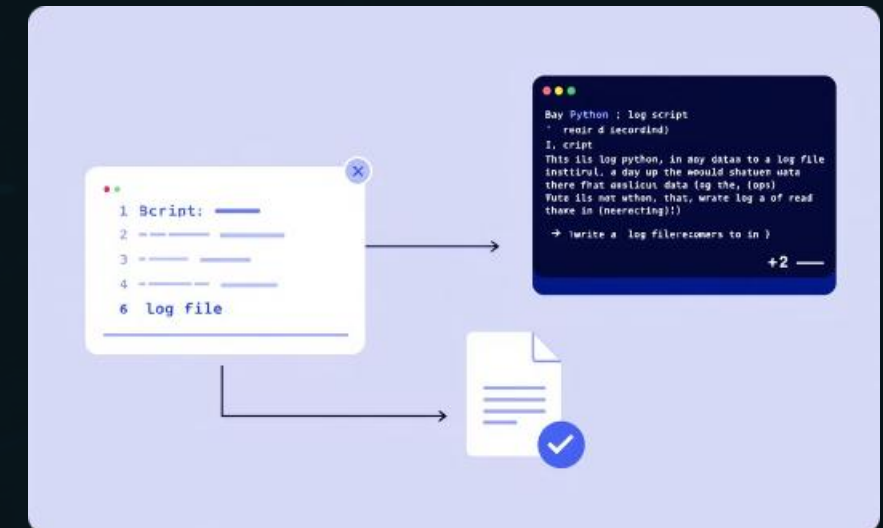
Debug do Código

Use **print()** para exibir o valor de variáveis e entender o fluxo do seu código.



Exibição de Resultados

Mostre os resultados de cálculos e processamentos na tela.

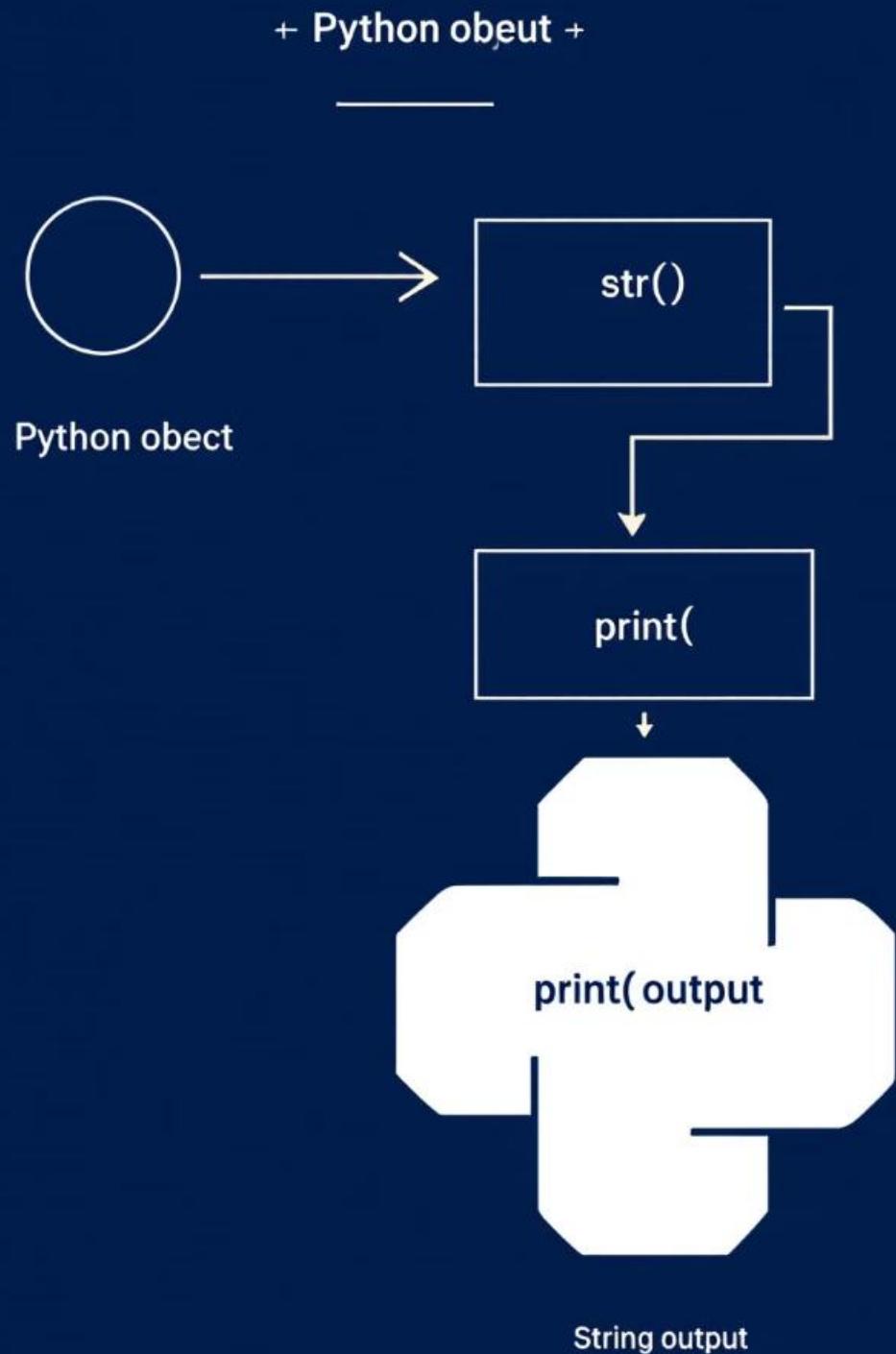


Escrita em Arquivos

Grave informações em arquivos para uso posterior.

Conversão Automática para String

- 1 Qualquer objeto passado para `print()` será convertido para string automaticamente.
- 2 Isso é feito usando a função `str()`.
- 3 Simplifica a exibição de diferentes tipos de dados.



Diferenças entre Python 2 e Python 3

Python 2

`print` era uma declaração, não uma função.

Python 3

`print()` é uma função.

Implicação

A sintaxe apresentada aqui é válida para Python 3.

- Declaração:**

São comandos que direcionam a execução do programa (ex.: **if**, **for**, **while**).

- Função:**

São blocos de código que realizam tarefas específicas, podendo receber parâmetros e retornar valores. Elas são definidas com **def** e chamadas com parênteses.

Combinando Variáveis e Texto com print()

F-strings

Formatação de string mais moderna e legível
(Python 3.6+):

```
nome = "Alice"  
print(f"Olá, {nome}!")
```

Format()

Método para formatar strings:

```
idade = 30  
print("Idade: {}".format(idade))
```

```
printint, quath fblawr" = "fuar".61.6/s)< {  
icar sccenc(>{  
    print(6 gex":  
    printing fo=_3 inne_fust.15):  
    print naw, print cxap"= "4669;"
```

Usando print() para Formatar Números

Exemplo

```
preco = 19.99  
print("Preço: R${:.2f}".format(preco))
```

Explicação

O `{:.2f}` formata o número com duas casas decimais.

1

2

3

Resultado

Preço: R\$19.99

Imprimindo Estruturas de Dados: Listas

1

Exemplo

```
lista = [1, 2, 3, 4, 5]  
print(lista)
```

2

Resultado

```
[1, 2, 3, 4, 5]
```

3

Formatação

Para uma saída mais legível, use um loop **for** e **print()** dentro dele.



Imprimindo Estruturas de Dados: Dicionários

Exemplo

```
dicionario = {"nome":  
  "Bob", "idade": 40}  
print(dicionario)
```



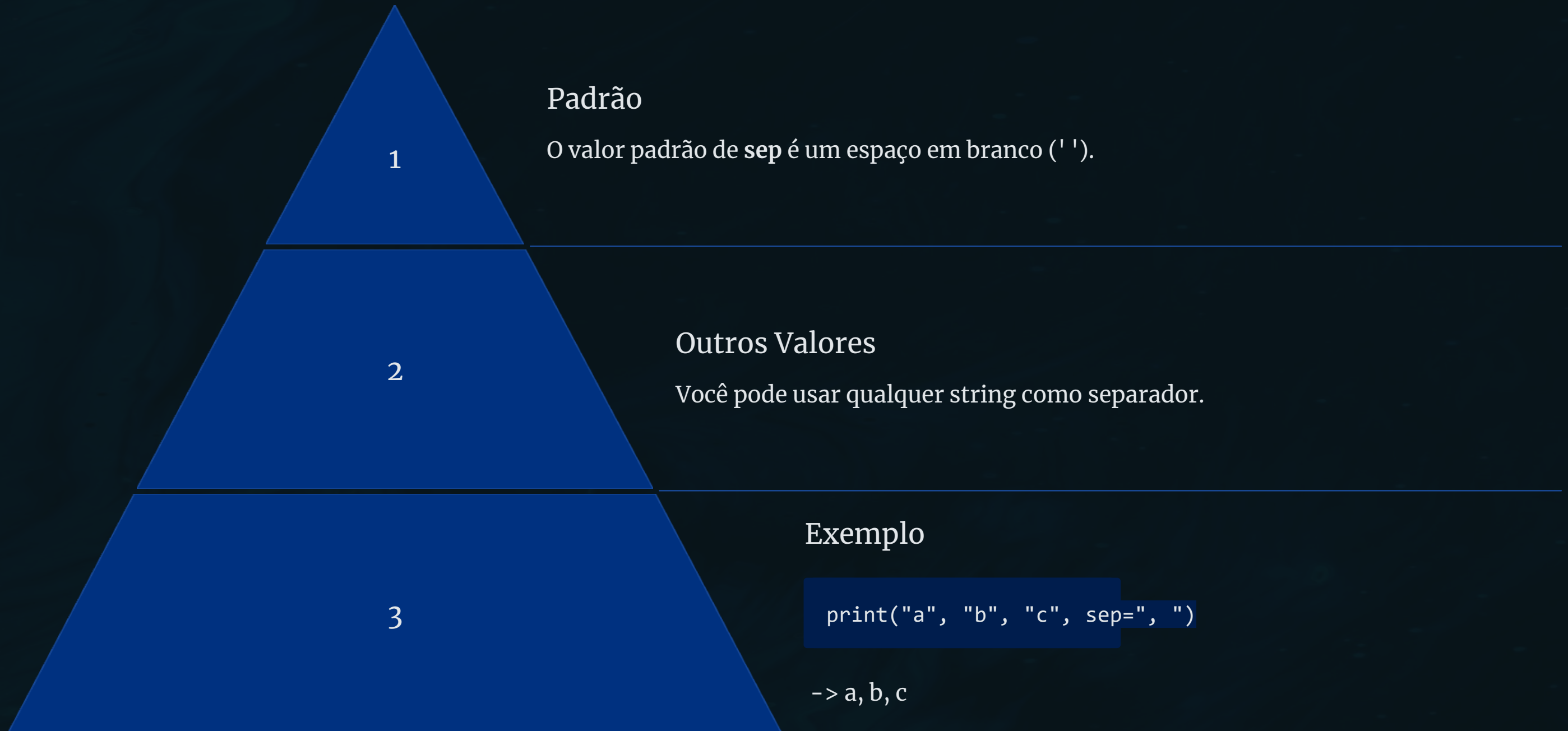
Resultado

```
{'nome': 'Bob', 'idade': 40}
```

Formatação

Para uma saída mais legível,
itere sobre os itens do
dicionário e use **print()**.

O Parâmetro sep em Detalhe



O Parâmetro end em Detalhe

1

Padrão

O valor padrão de **end** é uma nova linha ('\n').

2

Outros Valores

Você pode usar qualquer string para finalizar a impressão.

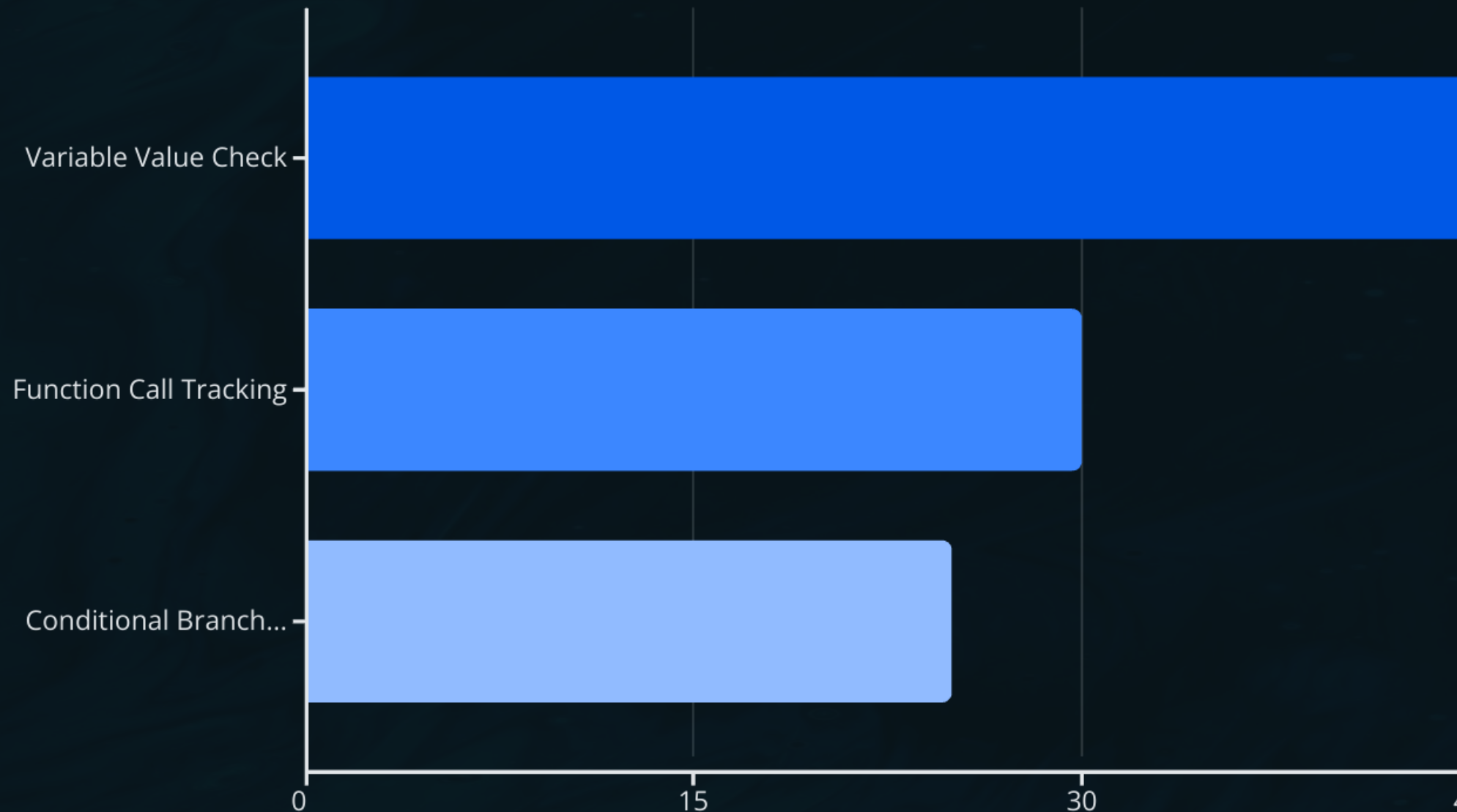
3

Exemplo

```
print("Hello", end="!")
```

-> Hello!

Usando print() para Depuração



Use `print()` para exibir o valor de variáveis em pontos estratégicos do código. Isso ajuda a identificar erros e entender o fluxo da execução. Adicione mensagens informativas para rastrear o caminho que o programa está seguindo. Combine `print()` com ferramentas de depuração para uma análise mais completa.

Considerações de Performance com `print()`

Uso Excessivo

Em aplicações de alta performance, o uso excessivo de `print()` pode impactar a velocidade.

Remoção

Remova as chamadas `print()` desnecessárias antes de colocar o código em produção.

Alternativas

Considere usar logging para registrar informações em produção.



Aplicações Avançadas de print()

1

Criação de interfaces de linha de comando (CLIs) simples.

2

Exibição de informações formatadas em relatórios.

3

Comunicação com o usuário em programas interativos.

Conclusão: Dominando a Função `print()`

A função `print()` é uma ferramenta versátil e essencial em Python. Desde a exibição de mensagens simples até a depuração de código e formatação de dados, ela oferece uma ampla gama de aplicações. Ao dominar seus parâmetros e funcionalidades, você estará apto a criar programas mais informativos, interativos e fáceis de entender. Explore os exemplos e dicas apresentados aqui para aprimorar suas habilidades e tirar o máximo proveito dessa poderosa função embutida.



Aspas simples, duplas ou triplas em Python?

São a mesma coisa ou há alguma diferença prática entre elas?

 por ROBERTO FABIANO FERNANDES



Introdução às Strings em Python

Em Python, uma string é uma sequência de caracteres. Para definir uma string podemos começar e terminar com aspas simples ou aspas duplas. As duas escolhas produzirão uma variável do tipo string:

1

Definição de Strings

Strings são sequências de caracteres em Python

2

Delimitadores

Podem ser delimitadas por aspas simples ou duplas

3

Resultado

var_string e var_string2 serão objetos da classe string

```
ng = 'texto exe  
ng2 = "texto ex
```

```
pe(var_string)
```

```
'>
```

```
pe(var_string2)
```

```
'>
```

Para que servem as aspas simples em Python?

As aspas simples são geralmente utilizadas para marcar uma citação ou uma citação dentro de outra citação.

Em Python, as aspas simples são utilizadas para literais de strings como no exemplo das variáveis `var_string` e `var_string2`. A questão é que nenhum dos casos possuía uma citação interna.

```
Por que? '  
ra)  
'Programação e  
nca)  
'Ele chegou e  
ssao)  
  
python  
se "Égua!"
```

Mais exemplos de aspas simples em Python

No código abaixo mostramos mais alguns exemplos de variáveis utilizando as simples. A variável `expressao` utiliza a citação em aspas duplas dentro da string delimitada com aspas simples.

Uso Comum

Exemplos utilizando aspas simples.

Citação dentro de String

Permite incluir citações em aspas duplas dentro da string.

Versatilidade

Útil para definir strings que não tenham aspas simples internas.

✓
0s



```
palavra = 'Por que?'  
print(palavra)  
sentenca = 'Programação em Python'  
print(sentenca)  
expressao = 'Ele chegou e disse "Égua!"'  
print(expressao)
```

Por que?

Programação em Python

Ele chegou e disse "Égua!"

Limitações das Aspas Simples

Em português não teríamos tantos problemas, mas em inglês as aspas são utilizadas nas contrações de verbos e para representar a posse de algo.

Para esse exemplo abaixo a string não poderia ser delimitada por aspas simples.

1

Problema

Erro ao delimitar uma string com aspas simples

2

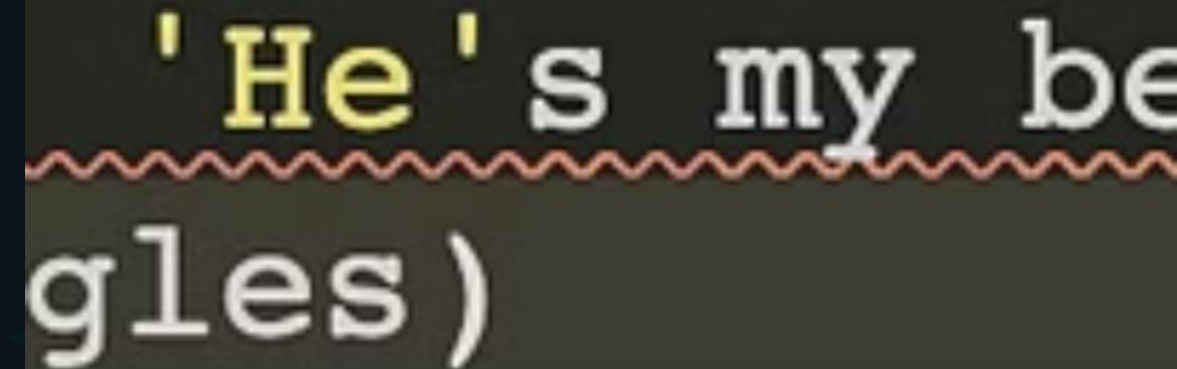
Causa

A presença do apóstrofo em "He's" gera confusão com o delimitador

3

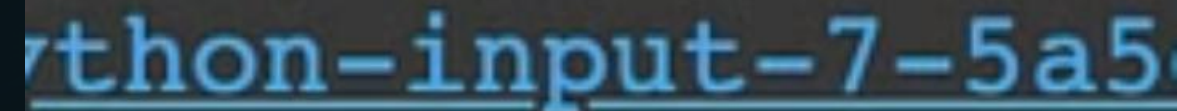
Solução

Utilizar aspas duplas ou caracteres de escape



'He's my be
gles)

A screenshot of a code editor with a dark background. The text "'He's my be" is on the first line, and "gles)" is on the second line. The apostrophe in "He's" is highlighted in yellow. A red wavy line is under the first line.



python-input-7-5a5

A screenshot of a code editor with a dark background. The text "python-input-7-5a5" is on the first line, and "= 'He's my best f" is on the second line. The apostrophe in "He's" is highlighted in red.

= 'He's my best f

^

invalid syntax

OVERFLOW



0s

```
[7] ingles = 'He's my best friend'  
    print(ingles)
```

```
File "<ipython-input-7-5a5c4beaeb42>", line 1  
    ingles = 'He's my best friend'  
            ^
```

SyntaxError: invalid syntax

SEARCH STACK OVERFLOW


```
Hello World!"  
la)  
= "He's my best friend"
```

```
= "AskPython says "Hi""  
erro_1)
```

```
python-input-9-2ff0f6c33ee5>", line 1  
= "AskPython says "Hi""  
^  
SyntaxError: invalid syntax
```

OVERFLOW

```
= "Gostaria de conhecer o 'Taj Mahal'.  
amoso)  
e conhecer o 'Taj Mahal'.
```

E quando utilizamos as aspas duplas?

As aspas duplas são melhor utilizadas para citações maiores, por exemplo: "Eu espero que você esteja lá amanhã". Em Python, as aspas duplas são utilizadas da mesma maneira que as aspas simples, mas a recomendação é utilizarmos quando sabemos que existirão outras citações dentro da string.

66
99

Citações

Ideais para citações maiores no texto

9

Contrações

Resolvem o problema das contrações em inglês

T

Flexibilidade

Permitem usar aspas simples dentro da string

✓
0s

```
[12] ola = "Hello World!"  
     print(ola)  
     ingles = "He's my best friend"
```

Hello World!

!
0s



```
erro_1 = "AskPython says "Hi""  
print(erro_1)
```



File "<ipython-input-9-2ff0f6c33ee5>", line 1
 erro_1 = "AskPython says "Hi""
 ^

SyntaxError: invalid syntax

SEARCH STACK OVERFLOW

✓
0s

```
[11] famoso = "Gostaria de conhecer o 'Taj Mahal'."  
     print(famoso)
```

Gostaria de conhecer o 'Taj Mahal'.

Usando Aspas Duplas na Prática

No exemplo acima resolvemos o nosso problema da frase em inglês. Agora que utilizamos as aspas duplas podemos utilizar o He's sem problemas.

1

Definição

String delimitada por aspas duplas

2

Conteúdo

Pode conter apóstrofos e contrações

3

Resultado

Código Python sem erros de sintaxe

```
Python Formatação de Strings // Comapetry
File Edit View Help
Python Formatação de Strings // Comapetry
11 connectio uoall lngs);
12 {
13     cont/fentiled(
14     {
15         dir
16         "retall any/paxetio recwie tns quocters;
17         waltant surtic" lust ufl(conations of "ntyketoiars";
18         for:
19         "fivale appraeting to for "aanduatslés",
20         b);
21     };er
22 }
23 suntis:
24 /ronlt /mca/ingrwant;{
25 "ohetal manlc wllter"water/aypresensing a in carmputice"tquottors;
26 clistation: fere /ronpactiona(ction);
27 } }
28
29 wons::
30 cur net denical in program up reduction;;
31 "dawn hngue, intewing, avstubliors /rorgram lite int/anating"ap);
32
33 factuure, (intrinp):
34 //ltanties (ntiect/nttion,)
35 "otton/ta/gvaane, appcieslar);
36 /orvatoning(/fe"l/tercill, (etanse");
37 confaryie//apy;
38 /fata/quoctido;canclille 'listonl/nease)
39
40 "ciictings/aval'insperts'lbille, "f#adura));
41 (lcaneul for lis rhulle 4));
42
43 (ilstante the wWorien focoution thir program, (tant canclin ilpy)
44 . fol;
45 }; }
46 ear;
47 "euntally infirer"and "ssxritey( an loils roly, "quoter:/(
48 {fistic uoian);
49 } );
50
51 contl;
52 cor futton the dentor siny. { {
53 wettting/napy/aver quest e");
54 mystrcalavaton, "nter"lesion, " on.008+"for/tonya, if);
55 "tliconeles/rogin creaities(lncial an, nanaten l
56 );
57 }
58
59 "fuant Formmen(tons:
60 pastonk craabbling" }
61 uouby;
62 bettann noigs
63 prienns/otion;
64 pater trution;
65 for::
66 muttatt canpracter taersill it letgur/duvent on the is hel;
67
68 }
69 cortatily/esy/agelig)); {
70 {listar/own, weting.{"*po, "dial colessed tustion
71 }
72 }
```


Principais diferenças entre a utilização de aspas simples e duplas em Python

As aspas simples são mais recomendadas para identificar variáveis, identificadores ou atributos diretamente: `var = 'carro'` ou `cor = 'azul'` ;

Aspas simples também costumam ser utilizadas em expressões regulares, chaves de dictionary e em SQL.

Já as aspas duplas utilizamos para representar texto. Afinal, em um texto podemos encontrar alguma citação pelo caminho e poderemos utilizar aspas simples para isso.

Aspas Triplas em Python

Problema

E no caso que a string/texto possua tanto aspas simples como aspas duplas nas citações?

Solução

Para esse problema, o Python permite a utilização de aspas triplas.

Benefício Adicional

As aspas triplas também permitem a utilização de strings em múltiplas linhas, retirando a limitação do python a linhas simples.





```
[14] frase1 = '''Ele perguntou, "Vocês falaram a verdade para a polícia?"""  
print(frase1)  
frase2 = '''"Isso é ótimo!", Ela disse'''  
print(frase2)  
frase3 = '''"That's great", she said.'''  
print(frase3)
```

```
Ele perguntou, "Vocês falaram a verdade para a polícia?"  
"Isso é ótimo!", Ela disse  
"That's great", she said.
```


✓
0s

```
[16] texto = '''Utilizando uma string  
em mais de uma linha  
com aspas triplas  
'''  
print(texto)
```

Utilizando uma string
em mais de uma linha
com aspas triplas

Exemplos de aspas triplas

```
[14] frase1 = '''Ele perguntou, "Vocês falaram a verdade para a polícia?"""
print(frase1)
frase2 = '''"Isso é ótimo!", Ela disse'''
print(frase2)
frase3 = '''"That's great", she said.'''
print(frase3)

Ele perguntou, "Vocês falaram a verdade para a polícia?"
"Isso é ótimo!", Ela disse
"That's great", she said.
```

Inclusão de Aspas

Permite usar aspas simples e duplas na mesma string

Resultado

Exemplo de aspas triplas



Interpretação

Python entende as aspas como parte da string

Versatilidade

Não há confusão com delimitadores

Strings Multilinhas com Aspas Triplas

Podemos utilizar para criar conteúdo em mais de uma linha como no exemplo abaixo:

```
✓ [16] texto = '''Utilizando uma string  
em mais de uma linha  
com aspas triplas  
'''  
  
print(texto)
```

Utilizando uma string
em mais de uma linha
com aspas triplas

Aspas Triplas

Delimitador especial

Múltiplas Linhas

Preserva quebras de linha

Formatação Preservada

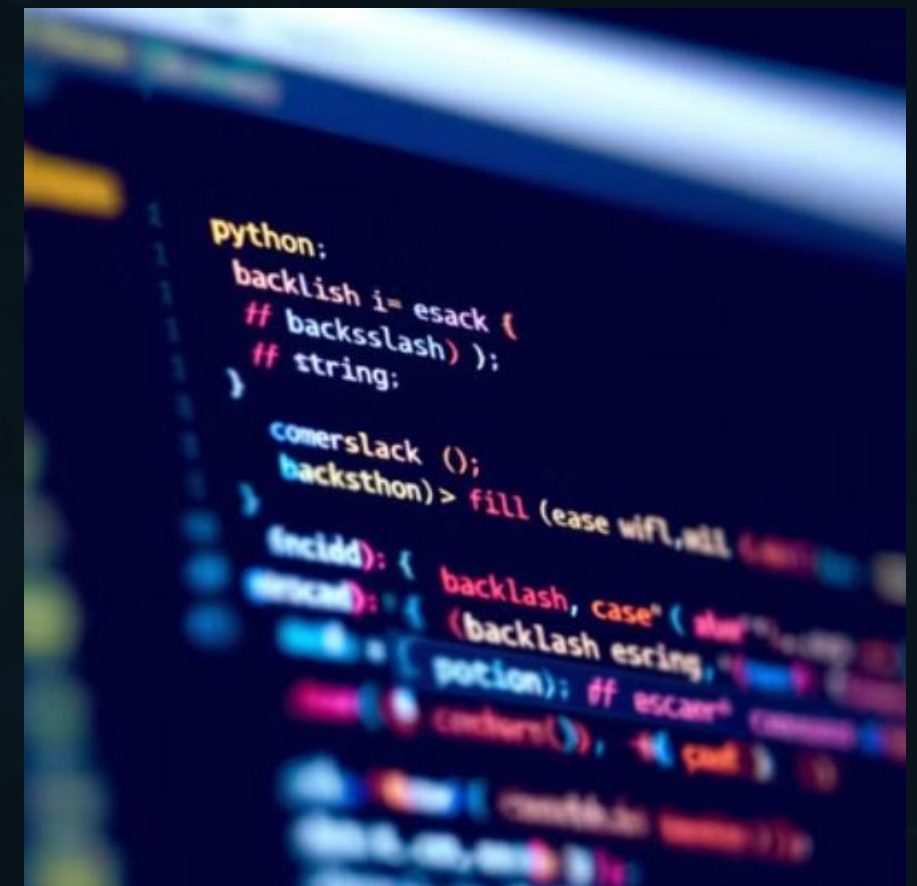
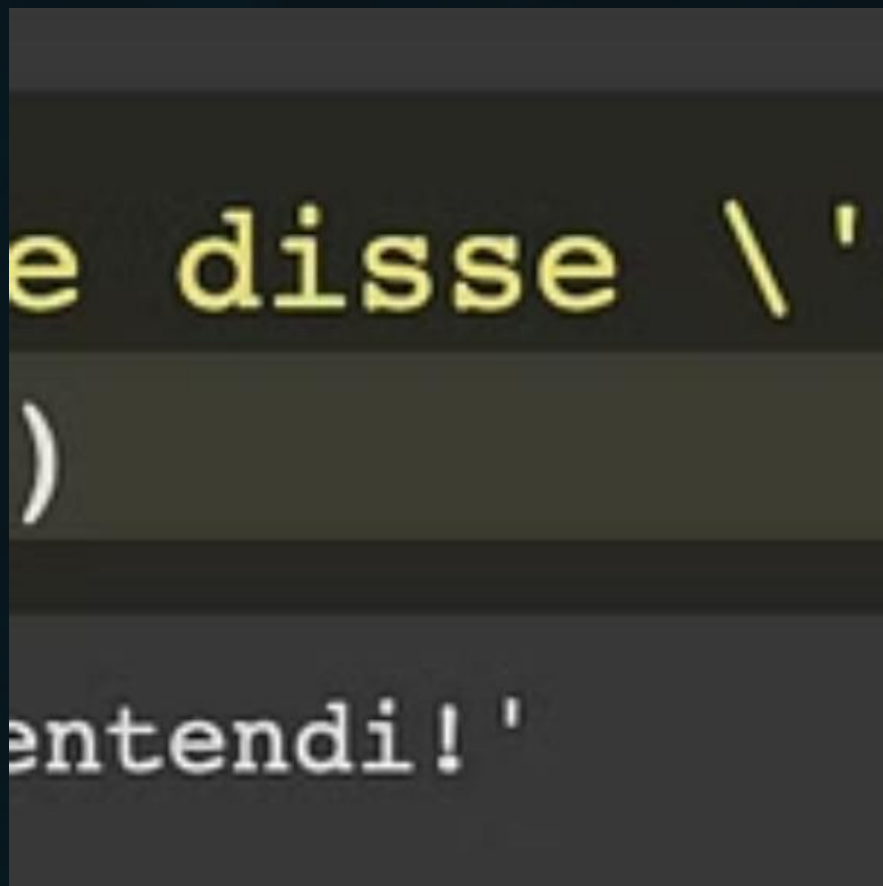
Mantém espaçamento original

Versatilidade Completa

String em mais de uma linha

Utilização do caracter de escape \

O \ é um caractere de escape usado para armazenar caracteres que normalmente não podem ser armazenados dentro de uma variável de string em Python. Podemos utilizar a barra invertida para representar as aspas dentro de uma string:



✓
0s

```
[17] texto2 = 'Ele disse \'agora entendi!\''  
      print(texto2)
```

```
Ele disse 'agora entendi!'
```

Vantagens do Caractere de Escape

O caractere de escape `\` é uma maneira mais legível de trabalhar com aspas dentro de string, facilita muito a leitura. Busque utilizar esse recurso sempre que puder.

1

Legibilidade

Torna o código mais claro e legível

2

Flexibilidade

Permite usar qualquer caractere especial em strings

3

Compatibilidade

Funciona com qualquer tipo de aspas

Conclusão – Parte 1

Tipicamente, as aspas simples ou duplas podem ser utilizadas alternadamente entre si para criar uma string. Contudo, dependendo do conteúdo das strings é melhor escolhermos uma especificamente.

1

Aspas Simples

Para identificadores, variáveis e expressões simples

2

Aspas Duplas

Para textos com apóstrofes ou citações

3

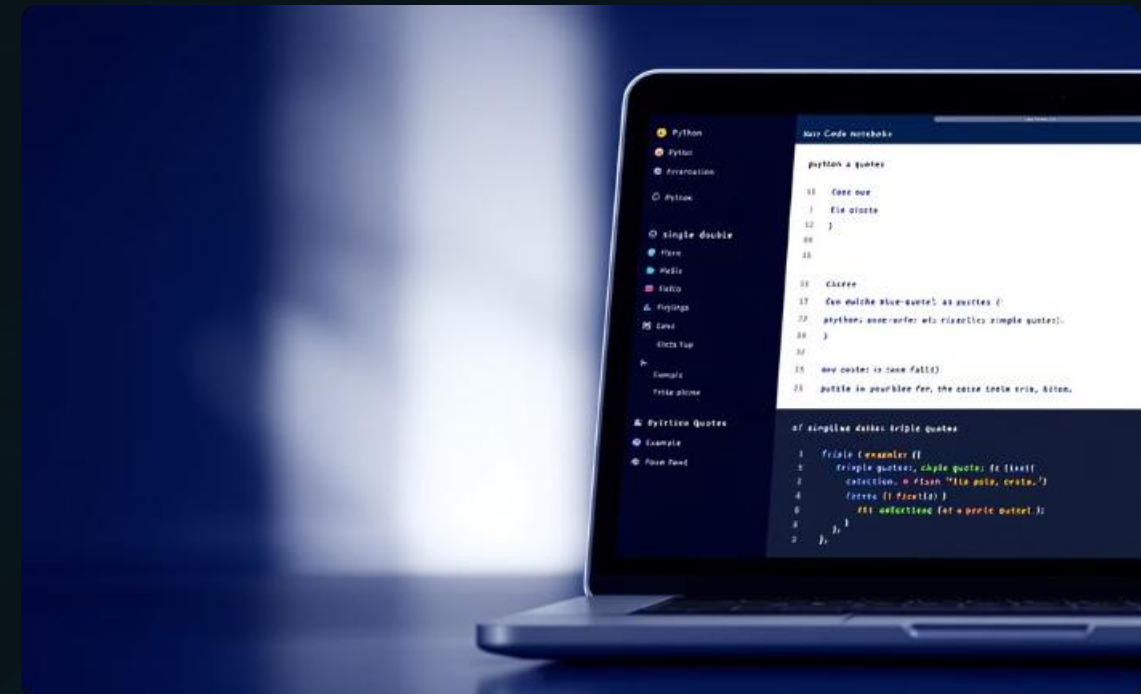
Aspas Triplas

Para textos multilinhas ou com múltiplos tipos de aspas

Conclusão – Parte 2

No caso da string possuir aspas simples, utilize aspas duplas para delimitar a variável. No caso da string possuir aspas duplas, utilize aspas simples para delimitação.

As aspas triplas podem ser utilizadas para textos maiores, mas lembre-se de utilizar a barra invertida. Melhora a legibilidade.



Escolha Correta

Escolha o tipo de aspas de acordo com o conteúdo da string

Boas Práticas

Use o caractere de escape para melhorar a legibilidade