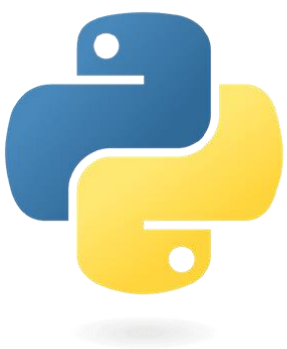
Manipulação de Strings em Python



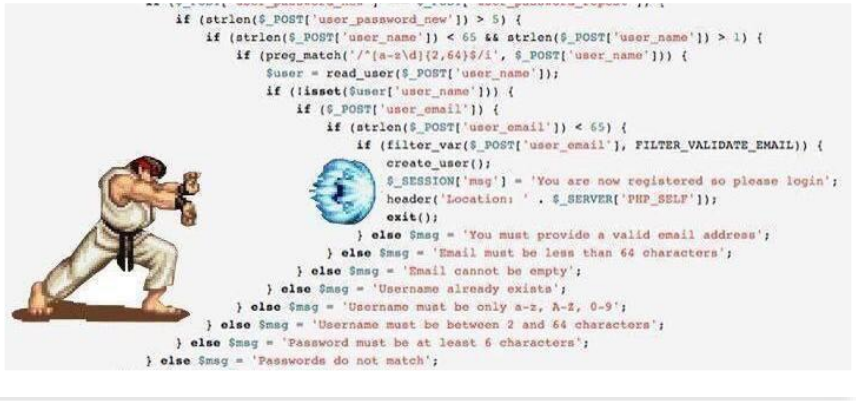
python™

Recife
2024

Indentação

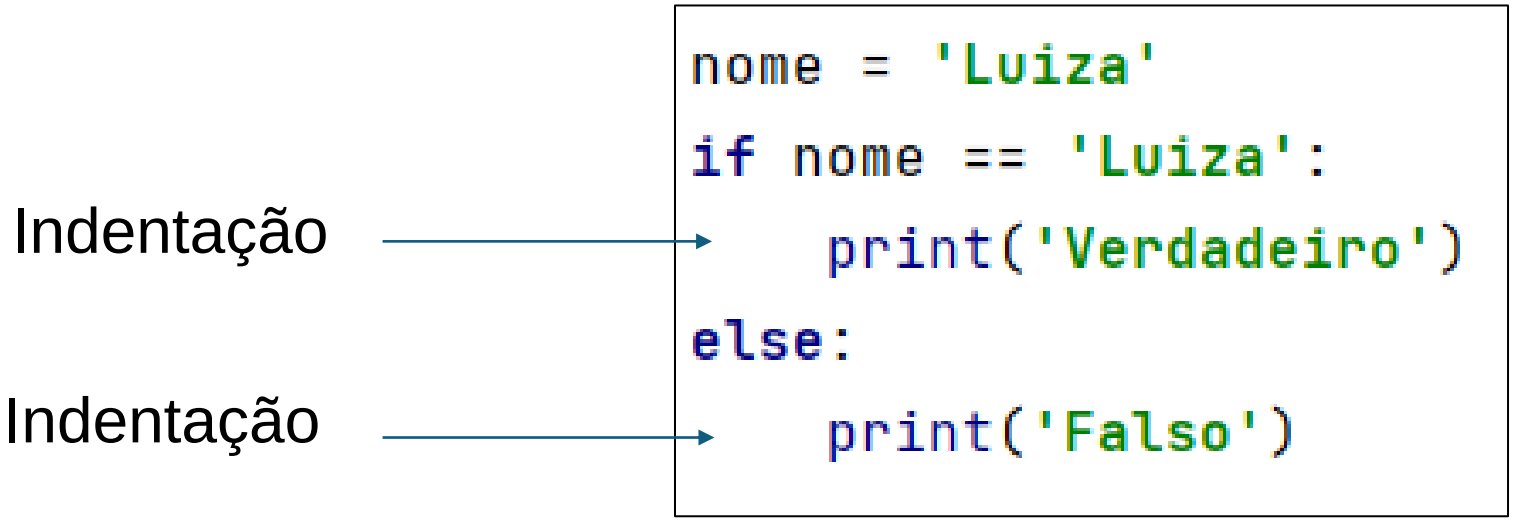
Antes de continuarmos e vermos um pouco mais sobre Python, precisamos entender um conceito muito utilizado em todas as linguagens de programação, e que em Python é obrigatório utilizar. Este conceito de organização das informações e do blocos adicionados ao Algoritmo é a **Indentação**.

Indentação são blocos de códigos que apresentam na sua estrutura um recuo, formando uma hierarquia em que o código será executado. Através do recuo que fazemos o alinhamento dos códigos, usa-se a **tabulação** ou **barra de espaço 4 vezes** para indicar o recuo.



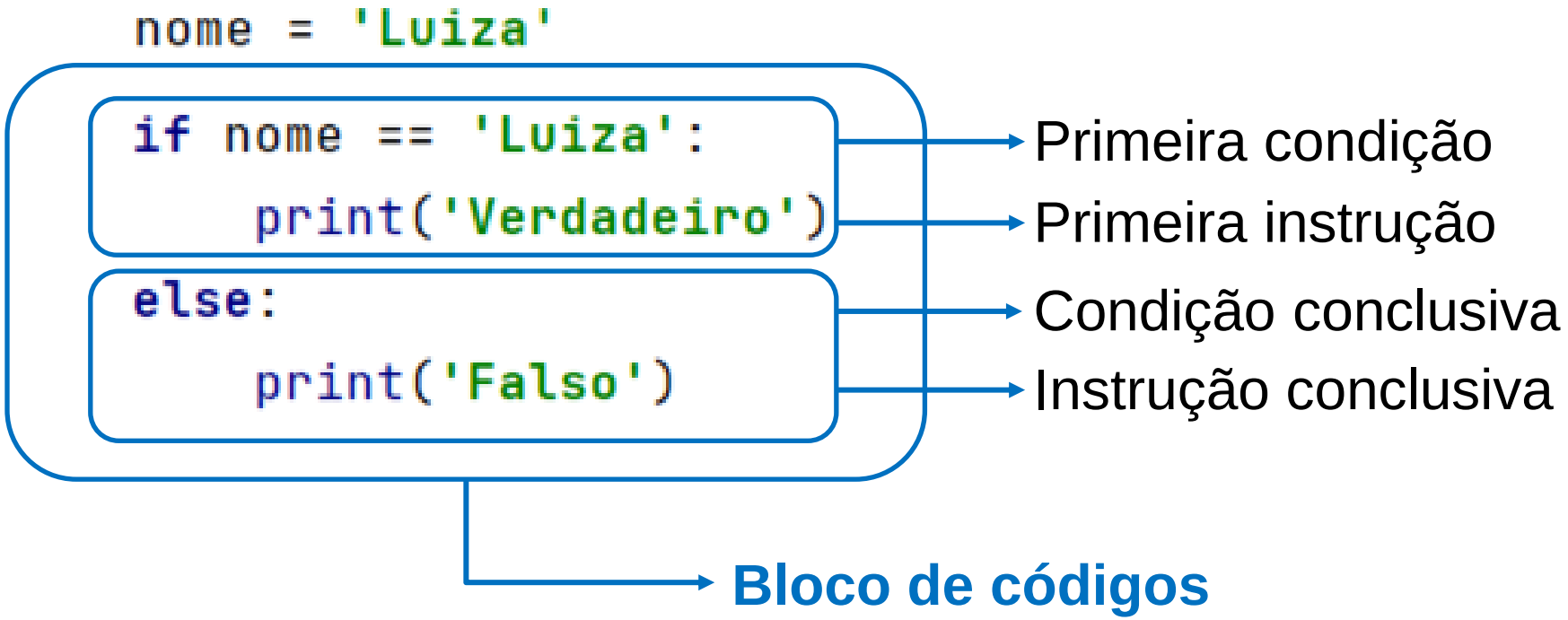
Indentação

Em Python a indentação é uma das principais características da sintaxe da linguagem.



Por que a indentação é importante no Python?

A indentação é importante porque serve para indicar o início e o fim de um bloco de códigos.



A indentação deixa o código mais legível e fácil de compreender

Exemplo de código sem indentação

```
numero = 12
if numero == 15:
print('Verdadeiro')
else:
print('Falso')
```

No Python nem conseguiria compilar esse código, pois ele é rigoroso com indentação.

```
numero = 12
if numero == 15: print('Verdadeiro')
else: print('Falso')
```

Neste caso, o Python reconhece as instruções mesmo sem indentações. O motivo é simples, a linguagem permite a condição e instrução em uma só linha.

Declarações - Statements

No Python a continuação em uma nova linha é permitido. Através do caracter \

```
total = 4 + \
      7 + \
    5
print(total) # 16
```

```
nome = 'maria' , \
         'Luiza' , \
         'Paulo'
print(nome)
```

Fonte: https://pythoniluminado.netlify.app/sintaxe

Palavras-chave em Python

Palavras Reservadas na linguagem

break	para sair de um loop
class	define uma classe
continue	continua para a nova iteração do loop
def	define uma função
del	deleta um objeto
elif	usado em comandos condicionais, como else e if
else	usado em comandos condicionais
except	usado com exceções, para tratar possíveis erros
False	Valor booleano, resulta de operações de comparação

Fonte: https://pythoniluminado.netlify.app/sintaxe

Palavras Reservadas na linguagem

finally	utilizado com **exceções**, um bloco de código que executará independente de ter uma exceção ou não
for	usado para criar um **loop**
from	para importar partes específicas de um **módulo**
global	declara uma **variável global**
if	usado para **comandos condicionais**
import	usado para importar **módulos**
in	capaz de checar se um valor está presente em uma lista, tupla, etc
is	testa se duas **variáveis** são iguais
lambda	cria uma **função anônima**
None	representa um valor **null**

Fonte: https://pythoniluminado.netlify.app/sintaxe

Palavras Reservadas na linguagem

not	operador lógico de negação
or	operador lógico "ou"
pass	comando null, um comando que não faz nada
raise	dispara um exceção
return	para sair de uma função e retornar um valor
True	Valor booleano, resulta de operações de comparação
try	Comando de try, usado em conjunto com except
while	Cria um loop while

Fonte: https://pythoniluminado.netlify.app/sintaxe

Strings em Python

Strings em Python

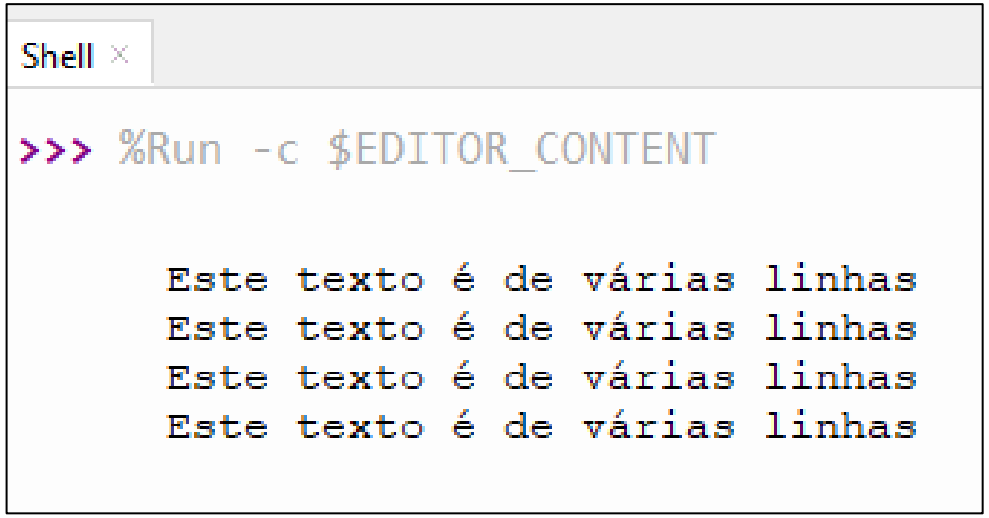
Para strings com várias linhas, utilize 3 aspas e uma variável para guardar o valor:

Strings com várias linhas

```
texto = '''
    Este texto é de várias linhas
    Este texto é de várias linhas
    Este texto é de várias linhas
    Este texto é de várias linhas
   '''

print(texto)
```

Console

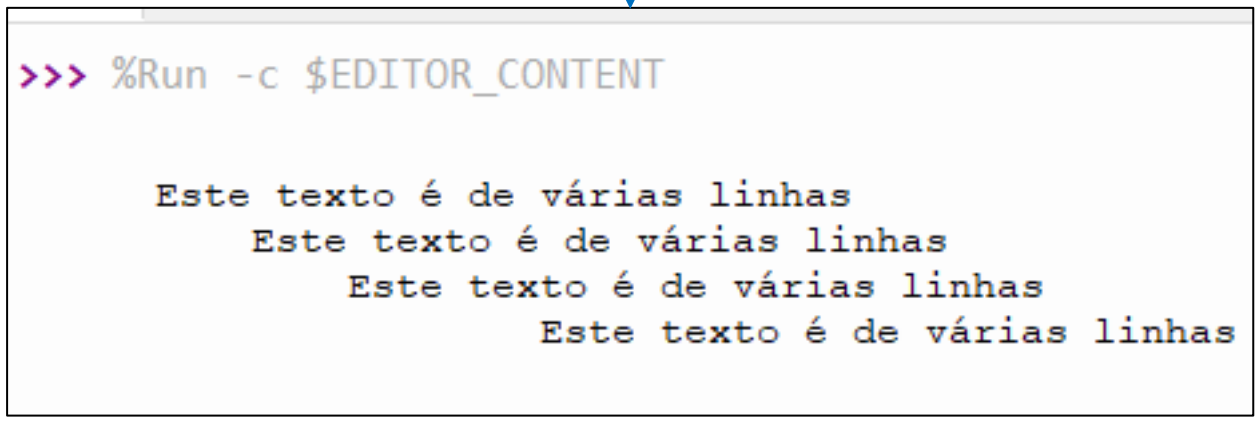


Observação

Do jeito que você definir o texto dentro das aspas, será mostrado na tela.

```
texto = '''
    Este texto é de várias linhas
        Este texto é de várias linhas
            Este texto é de várias linhas
                Este texto é de várias linhas
    '''

print(texto)
```



Manipulação de Strings

Strings, também chamados de cadeias de caracteres. No Python possui várias operações associadas. Vejamos algumas delas a seguir:

Saber o tamanho de uma String

a = ‘João’
print(len(a))

len pega o tamanho da variável

Resultado: 4

```
frase = 'Curso de Programação de Sistemas'
print(len(frase))
```

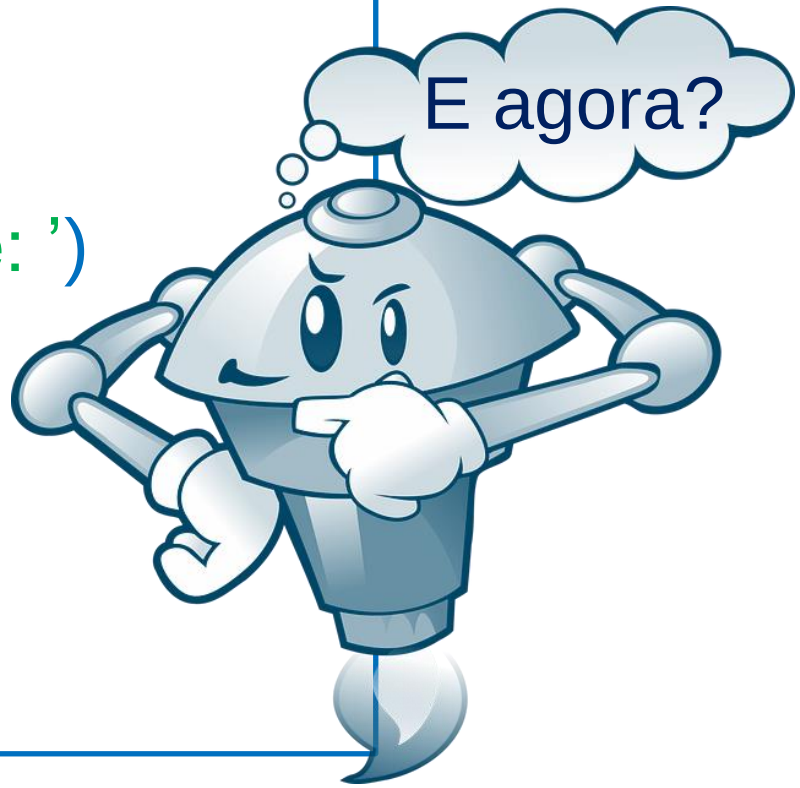
Resultado: 32

Concatenando uma String

Juntando Strings

```
nome = input('Informe seu nome: ')
sobrenome = input('Informe seu sobrenome: ')
resultado = nome + sobrenome
print(resultado)
```

Console: **JoãoPaulo**



Resolução

Juntando Strings

```
nome = input('Informe seu nome: ')
sobrenome = input('Informe seu sobrenome: ')

resultado = nome + ' '+ sobrenome
print(resultado)
```

Console: **João Paulo**

String de Associação

Para verificar se uma determinada frase ou caractere está presente em uma string, podemos usar a palavra-chave in.

```
frase = 'Curso de Programador de Sistemas'
print('Programador' in frase)
```

Se existe retornará TRUE

String de Associação

Verificar se não existe

```
frase = 'Curso de Programador de Sistemas'
print('Programador' not in frase)
```

Retornará FALSE, pois existe

Modificando Strings

Python tem um conjunto de métodos integrados que você pode usar em strings.

O método upper() retorna a string em maiúsculas.

```
texto = ‘Texto exemplo’
print(texto.upper())
```

Console:
TEXTO EXEMPLO

O Método lower() retorna a string em letras minúsculas.

```
texto = ‘Texto exemplo’
print(texto.lower())
```

Console:
texto exemplo

Modificando Strings

O método capitalize() retorna a primeira letra em maiúscula e o restante fica minúscula.

```
texto = ‘exemplo TESTE’
print(texto.capitalize())
```

Console:
Exemplo teste

O método split (‘’) método retorna uma lista onde o texto entre o separador especificado se torna os itens da lista.

```
texto = ‘banana uva limão, morango’
print(texto.split(‘,’))
```

Console:
['banana uva limão', ' morango']

Modificando Strings

O método strip() remove qualquer espaço em branco do início ou do fim;

lstrip remove somente a esquerda.

rstrip remove somente a direita.

```
nome = ‘      Maria’
print(nome.strip())
```

Console:
Maria

O método replace(‘’, ‘’) substitui uma string por outra string.

```
frase =‘Texto exemplo’
print(frase.replace(‘Texto’, ‘teste’))
```

Console:
teste exemplo

MÉTODO	COMO FUNCIONA	DESCRIÇÃO
center()	print(var.center(100))	Imprime na tela com espaçamento de 100 caracteres
count(‘ ’)	print(var.count(‘banana’))	Retorna o número de vezes que o valor ‘banana’ aparece na string
Endswith(‘ ’)	print(var.endwith(‘.’))	Retorna verdadeiro ou falso se a string termina com ponto final.
isalnum()	print(var.isalnum())	Verifica se a string tem alfanuméricos, ou seja, letra do alfabeto (az) e números (0-9).
isalpha()	print(var.isalpha())	Verifique se todos os caracteres do texto são letras
isidentifier()	Print(texto.isidentifier())	Verifica se o texto digitado é uma variável válida

MÉTODO	COMO FUNCIONA	DESCRIÇÃO
islower())	print(var.islower())	Verifica se todos os caracteres do texto estão em letras minúsculas
isnumeric()	print(var.isnumeric())	Verifica se todos os caracteres são números
isupper()	print(var.isupper())	Verifique se todos os caracteres do texto estão em maiúsculas
startswith(‘’)	print(var.startswith(‘’))	método retorna True se a string começar com o valor especificado, caso contrário, False.
swapcase()	txt.swapcase()	Deixe as letras que são minúsculas em maiúsculas e vice - versa

Exemplos

Exemplos

Usando o comando replace

```
frase = 'Curso de Programador de Sistemas'
print(frase.replace('Programador de Sistemas', 'Python'))
```

Console

```
Curso de Python
```

Exemplos

A contagem diferencia o maiúsculo do minúsculo.
Ex.: **p** = 0 // **P** = 1 // **S** = 1 // **s** = 3

```
frase = 'Curso de Programador de Sistemas'
print(frase.count('a')) ⟶ 3
print(frase.count('A')) ⟶ 0
print(frase.upper().count('A')) ⟶ 3
```

Exemplo

```
info = input('Digite uma informação: ')
print('A informação é do tipo?',type(info))
print('A informação é um número?',info.isnumeric())
print('A informação é um alfabético?',info.isalpha())
print('A informação é um alfanumérico?',info.isalnum())
print('A informação é um espaço?',info.isspace())
print('A informação é captalizada?',info.istitle())
print('A informação é maiúsculo?',info.isupper())
print('A informação é minúsculo?',info.islower())
```

```
Digite uma informação: Olá
A informação é do tipo? <class 'str'>
A informação é um número? False
A informação é um alfabético? True
A informação é um alfanumérico? True
A informação é um espaço? False
A informação é captalizada? True
A informação é maiúsculo? False
A informação é minúsculo? False
```

```
Digite uma informação: 233
A informação é do tipo? <class 'str'>
A informação é um número? True
A informação é um alfabético? False
A informação é um alfanumérico? True
A informação é um espaço? False
A informação é captalizada? False
A informação é maiúsculo? False
A informação é minúsculo? False
```

Fatiamento de Strings

Fatiamento de Strings

As Strings são iteráveis, ou seja, podemos pegar elementos específicos dentro de uma string.

Deste modo, surge o fatiamento que consiste em receber uma substring de uma determinada string, ou seja, partes especificas. Podemos definir o intervalo que desejamos receber.

Fonte: DOWNEY, 2016

Fatiamento Simples - Substrings das Strings

Podemos acessar caracteres específicos de uma String através de conotações [].
O primeiro caractere de uma String sempre começará no índice 0.

Buscando um caractere pelo seu índice

```
caractere = 'Substrings em Python'
print(caractere[0])
print(caractere[5])
print(caractere[19])
print(caractere[20])
```

Tamanho da String: 20
Índice da String: tamanho -1

Resultado: S

Resultado: r

Resultado: n

```
IndexError: string index out of range
```

Fatiamento de sequências

Para acessar fatias de uma String, podemos especificar a posição inicial e final da fatia.

Início: É o primeiro índice a ser considerado;

Fim – 1: É o último índice a ser considerado

	0	1	2	3	4	5
p =	P	Y	T	H	O	N
	-6	-5	-4	-3	-2	-1

Exemplo - Fatiamento de sequências

	0	1	2	3	4	5
p =	P	Y	T	H	O	N
	-6	-5	-4	-3	-2	-1

```
print('Tamanho:',len(p))
```
→ Tamanho: 6 Índice: 5

```
print(p[:5])
```
→ Começará do índice 0 até chegar no 5-1 **PYTHO**

```
print(p[2:6])
```
→ Começará do índice 2 até chegar no 6-1 **THON**

```
print(p[0:7])
```
→ Começará do índice 0 até chegar no 7-1 **PYTHON**

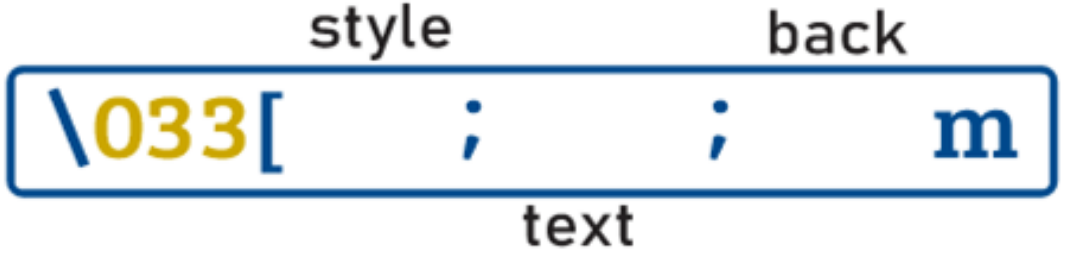
```
print(p[:])
```
→ Começará do índice 0 até chegar no último **PYTHON**

```
print(p[-3:-1])
```
→ Começará do índice -3 até chegar no -2 **HO**

```
print(p[-3:])
```
→ Começará do índice -3 até chegar no último **HON**

Aplicando CORES no python

Cores no Python



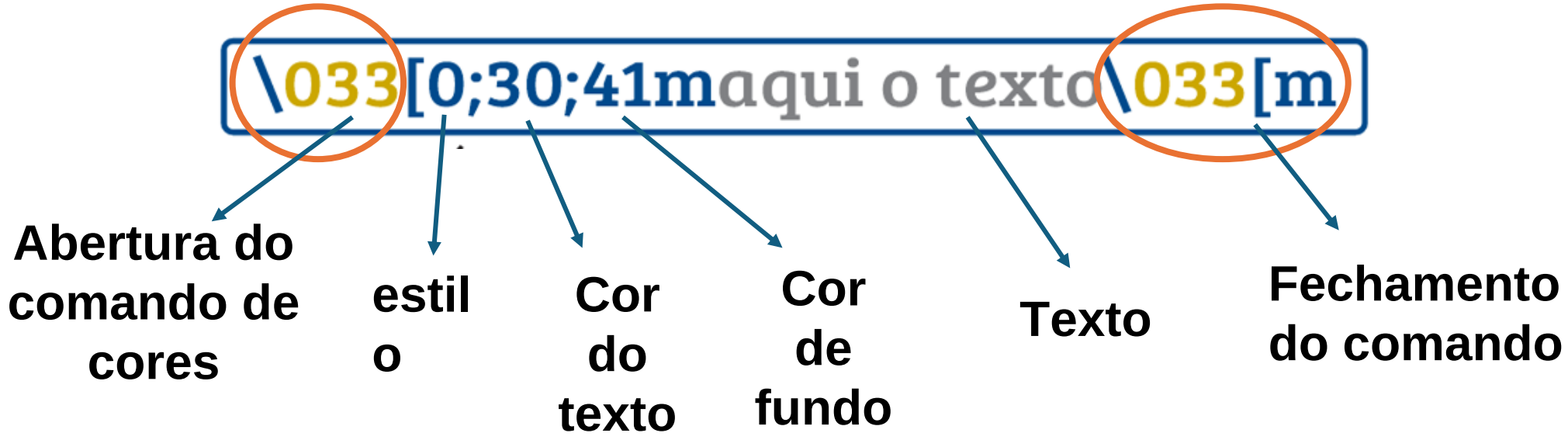

```
print('\033[0;30;41mOlá Mundo!\033[m')
```

```
Olá Mundo!
```

Tabela de Cores

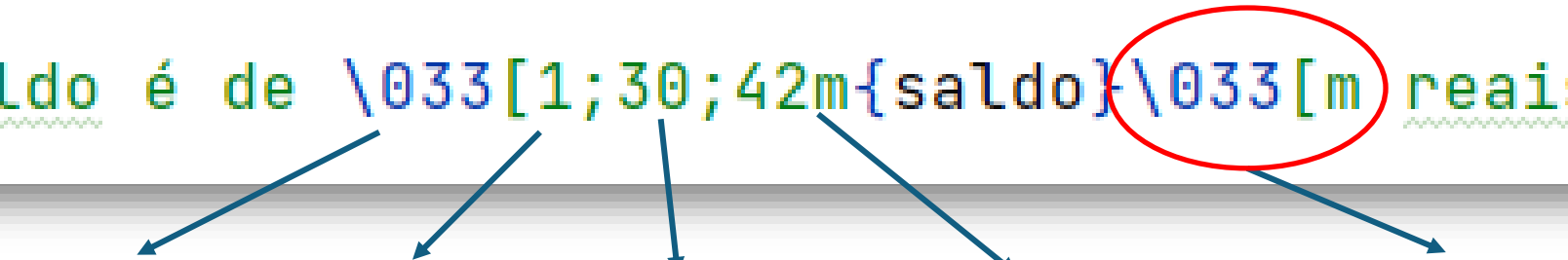
https://ansi.gabebanks.net/

Cores no Python



Cores no Python

```
saldo = 100
print(f'Seu saldo é de \033[1;30;42m{saldo}\033[m reais')
```




```
Seu saldo é de 100 reais
```

Exemplo 1 – Chá de Revelação

```
#Chá de revelação
a = 'menino'
b = 'menina'
```

Como utilizar as 4 formas de impressão na tela:

f-string

```
print(f'O sexo do bebê é \033[1;34m{a}\033[m ou \033[1;32m{b}\033[m?')
```

.format

```
print('O sexo do bebê é \033[1;34m{}\033[m ou \033[1;32m{}\033[m?'.format(a,b))
```

vírgula

```
print('O sexo do bebê é\033[1;34m',a,'\033[mou\033[1;32m',b,'\033[m?')
```

Concatenaçã o

```
print('O sexo do bebê é \033[1;34m'+ a +'\033[m ou \033[1;32m' + b +'\033[m?')
```

```
O sexo do bebê é menino ou menina?
```

Exemplo

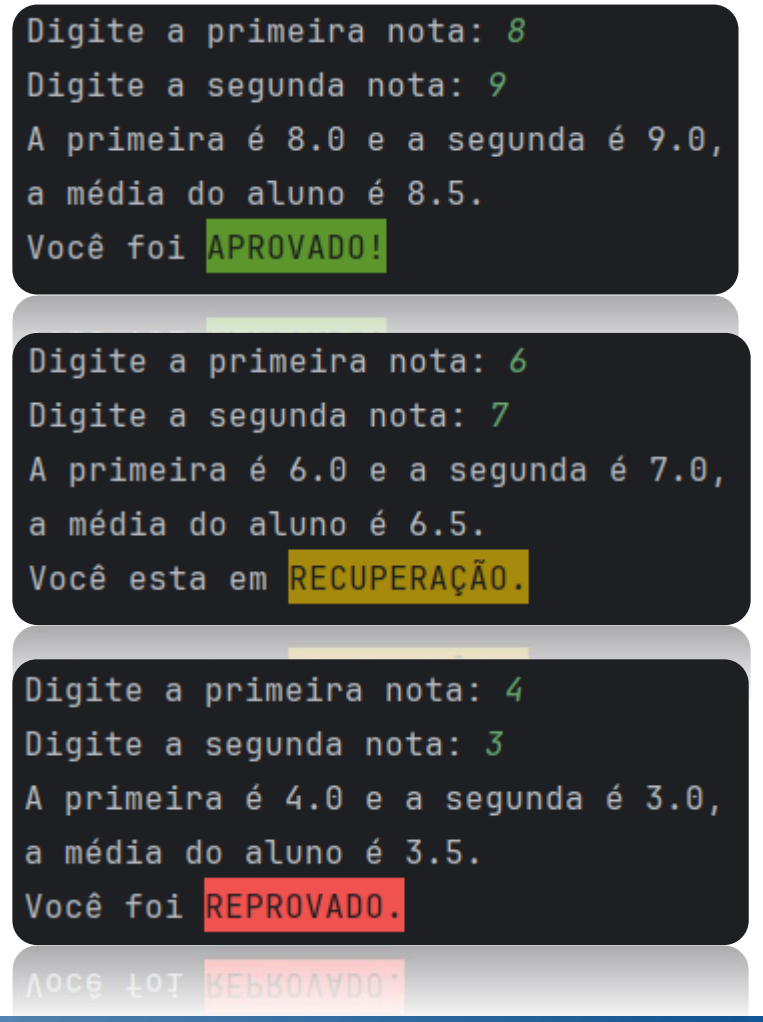
```
a = 'menino'
b = 'menina'
print('O sexo é \033[32m{}\033[m ou \033[35m{}\033[m?'.format( *args: a,b))
print('O sexo é {}{}{}!!!'.format( *args: '\033[7;0;45m',b,'\033[m'))
```

```
O sexo é menino ou menina?
O sexo é menina!!!
```

Exercício

Crie um programa que o usuário digite duas notas e apareça uma mensagem informando se ele foi:

- APROVADO: nota superior a 7.0
- RECUPERAÇÃO: nota entre 5.0 e 6.9
- REPROVADO: nota abaixo de 5.0



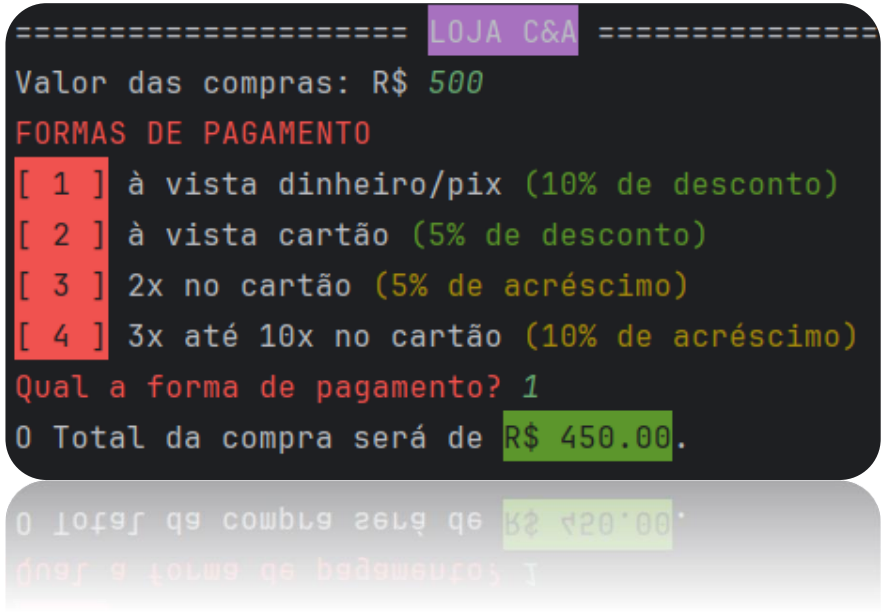
Exemplo

```
nota1 = float(input('Digite a primeira nota: '))
nota2 = float(input('Digite a segunda nota: '))
media = (nota1 + nota2)/2
print('A primeira é {:.1f} e a segunda é {:.1f}, '
      '\na média do aluno é {:.1f}.'.format( *args: nota1,nota2,media))
if media < 5:
    print('Você foi \033[7;31mREPROVADO.\033[m')
elif 7 > media >= 5:
    print('Você esta em \033[7;33mRECUPERAÇÃO.\033[m')
elif media > 7:
    print('Você foi \033[7;32mAPROVADO!\033[m')
```

Exercício

Crie um programa que o usuário calcule o valor a ser pago na loja de acordo com as condições de pagamento:

- À vista (dinheiro/pix): 10% de desconto
- À vista (cartão): 5% de desconto
- 2x no cartão: 5% de acréscimo
- 3x até 10x no cartão: 10% de acréscimo



Exemplo

```
====================  LOJA C&A  ===============
Valor das compras: R$ 500
FORMAS DE PAGAMENTO
[ 1 ] à vista dinheiro/pix (10% de desconto)
[ 2 ] à vista cartão (5% de desconto)
[ 3 ] 2x no cartão (5% de acréscimo)
[ 4 ] 3x até 10x no cartão (10% de acréscimo)
Qual a forma de pagamento? 1
O Total da compra será de R$ 450.00.
```

```
====================  LOJA C&A  ===============
Valor das compras: R$ 500
FORMAS DE PAGAMENTO
[ 1 ] à vista dinheiro/pix (10% de desconto)
[ 2 ] à vista cartão (5% de desconto)
[ 3 ] 2x no cartão (5% de acréscimo)
[ 4 ] 3x até 10x no cartão (10% de acréscimo)
Qual a forma de pagamento? 2
O Total da compra será de R$ 475.00.
```

```
====================  LOJA C&A  ===============
Valor das compras: R$ 500
FORMAS DE PAGAMENTO
[ 1 ] à vista dinheiro/pix (10% de desconto)
[ 2 ] à vista cartão (5% de desconto)
[ 3 ] 2x no cartão (5% de acréscimo)
[ 4 ] 3x até 10x no cartão (10% de acréscimo)
Qual a forma de pagamento? 3
O valor da parcela será 2x de R$ 262.50 com juros.
O Total da compra será de R$ 525.00.
```

```
====================  LOJA C&A  ===============
Valor das compras: R$ 500
FORMAS DE PAGAMENTO
[ 1 ] à vista dinheiro/pix (10% de desconto)
[ 2 ] à vista cartão (5% de desconto)
[ 3 ] 2x no cartão (5% de acréscimo)
[ 4 ] 3x até 10x no cartão (10% de acréscimo)
Qual a forma de pagamento? 4
Em quantas parcelas (de 3x até 10x)? 5
O valor da parcela será 5x de R$ 110.00 com juros.
O Total da compra será de R$ 550.00.
```