Aula 07

Revisando

Na aula anterior vimos como funcionam os blocos de códigos no Python e os comandos if...elif...else.

Os blocos de código são usados depois de certas palavras-chave, como if, else, while, for entre outras. Eles são usados para agrupar instruções que fazem parte da mesma estrutura lógica.

```
num = 10

if num > 0:
    print('esse número é maior do que zero')
else:
    print('esse número é menor do que zero')
```

Por enquanto, vimos apenas um grupo de comandos que fazem uso do bloco: if, elif e else.

Para um bloco de código do if ser executado, ele precisa receber um valor lógico True ou False. A forma que temos de chegar neles, é através do uso dos operadores condicionais. TODO teste de um operador condicional vai gerar um valor booleano True ou False.

```
10 == 10 : True
10 > 10 : False
10 >= 10 : True
10 < 10 : False
10 <= 10 : True
10 != 10 : False
```

É a partir desse resultado da expressão booleana que o bloco if e elif vai ou não ser executado.

```
# abaixo é o que escrevemos quando queremos
# realizar um teste para executar um bloco
# de codigo dentro do if
if 10 == 10:
    print('realmente 10 é igual a 10')

# depois da comparação ser testada, é retornado
# True ou False. Se for True, aquele bloco será
# executado. Se for False, o bloco será ignorado
if True:
    print('realmente 10 é igual a 10')
```

Os comandos elif e else dependem do if para serem usados, mas podemos fazer várias sequências de apenas if. O código abaixo, apesar de funcional, ele não é eficiente. Se já verificamos que um número é positivo, não há necessidade de verificar se ele também é negativo ou igual a zero. Estaremos fazendo 3 testes, quando que poderíamos ter realizado apenas 1.

```
num = 10

if num > 0:
    print('número positivo')
if num < 0:
    print('número negativo')
if num == 0:
    print('é zero')
```

Para arrumarmos isso, usamos os comandos elif e else em conjunto com o if.
Abaixo temos o mesmo código, mas agora ele está eficiente.

```
num = 10

if num > 0:
    print('número positivo')
elif num < 0:
    print('número negativo')
else:
    print('é zero')
```

Apesar dos códigos não serem muito diferentes visualmente, eles são muito diferentes no quesito de execução. Agora, o bloco do elif será testado APENAS se o resultado da condição if for falsa. Para o bloco else ser testado, é necessário que TODOS os resultados das condições anteriores sejam falsos.

Também podemos aninhar os comandos if um dentro do outro, criando blocos de código dentro de outros blocos de código.

```
num = int(input('digite um número : '))

# primeiro testa se é igual ou maior que zero
if num >= 0:
    # agora testa se é ímpar
    if num % 2 == 1:
        print('o num é ímpar positivo')
    # por eliminação, se o número não é ímpar
    # só pode ser par
    else:
        print('o num é par positivo')
# se não é igual ou maior que zero, então só
# pode ser negativo
else:
    print('num é um negativo qualquer')
```

Tipo Lista

Python inclui diversas estruturas de dados compostas, usadas para agrupar outros valores. A mais versátil é o tipo list (lista), que pode ser escrita como uma lista de valores (itens) separados por vírgula, entre colchetes. Os valores contidos na lista não precisam ser todos do mesmo tipo.

```
# uma lista de apenas inteiros
numeros = [1,2,3,4,5]

# uma lista de apenas strings
planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']

# uma lista de booleanos
resultados = [True, False, False, True]

# uma lista composta de vários tipos
misto = [42, 3.14, 'uma string', False, [1, 2, 3]]
```

Índices de uma Lista

Uma lista pode acessada através de seu índice. O índice de uma lista sempre será um número inteiro crescente iniciado em 0.

```
# uma lista de apenas strings
planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']

# mostra o tipo
print(type(planetas))
# mostra toda a lista de uma só vez
print('Lista de planetas :', planetas)
# mostra cada item individualmente de acordo
# com o índice
print('Planeta no índice 0 :', planetas[0])
print('Planeta no índice 1 :', planetas[1])
print('Planeta no índice 2 :', planetas[2])
print('Planeta no índice 3 :', planetas[3])
```

Quando usamos o índice de uma lista para acessar um determinado valor, é como se estivéssemos acessando uma variável diretamente. Por causa disso, podemos também alterar seus valores diretamente.

```
# uma lista de apenas strings
planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']

# mostra todos os planetas
print('Lista de planetas :', planetas)

# atribuindo um novo valor na posição 1
planetas[1] = 'Júpiter'
print('Lista de planetas :', planetas)
```

Se tentarmos acessar um índice que não existe na nossa lista, iremos levantar o erro `IndexError` (iremos ver mais sobre os erros adiante).

```python
# uma lista de apenas strings
planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']

print('Lista de planetas :', planetas)
print('Planeta no índice 10 :', planetas[10])
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

C:\Users\gutoh\Desktop>python teste.py
Lista de planetas : ['Mercúrio', 'Vênus', 'Terra', 'Marte']
Traceback (most recent call last):
  File "C:\Users\gutoh\Desktop\teste.py", line 5, in <module>
    print('Planeta no índice 10 :', planetas[10])
                                    ~~~~~~~~^^^^
IndexError: list index out of range
```

Se usarmos um valor negativo para acessar o índice, em vez de acessar ele do início para o fim, iremos acessar ele do fim para o começo.

```python
# uma lista de apenas strings
planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']

# mostra o tipo
print(type(planetas))
# mostra toda a lista de uma só vez
print('Lista de planetas :', planetas)
# mostra cada item individualmente de acordo
# com o índice
print('Planeta no índice -1 :', planetas[-1])
print('Planeta no índice -2 :', planetas[-2])
print('Planeta no índice -3 :', planetas[-3])
print('Planeta no índice -4 :', planetas[-4])
```

```
C:\Users\gutoh\Desktop>python teste.py
<class 'list'>
Lista de planetas : ['Mercúrio', 'Vênus', 'Terra', 'Marte']
Planeta no índice -1 : Marte
Planeta no índice -2 : Terra
Planeta no índice -3 : Vênus
Planeta no índice -4 : Mercúrio
```

Também podemos usar os colchetes para separar intervalos de uma lista.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> # mostrando todos os elementos da lista
>>> planetas[:]
['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> # pegando todos os elementos a partir do segundo
>>> planetas[1:]
['Vênus', 'Terra', 'Marte']
>>> # pegando todos os elementos até o terceiro (não inclui ele)
>>> planetas[:2]
['Mercúrio', 'Vênus']
```

Alguns Métodos de Listas

As listas diversos métodos para podemos manipular elas mais facilmente.
O método append vai adicionar um novo elemento ao final da lista, alterando a lista original.
O método pop vai remover o último elemento da lista e retornar ele, que pode ser salvo em uma variável.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> planetas
['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> planetas.append('Júpiter')
>>> planetas
['Mercúrio', 'Vênus', 'Terra', 'Marte', 'Júpiter']
>>> removido = planetas.pop()
>>> planetas
['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> removido
'Júpiter'
```

O método pop também pode ser usado passado um número inteiro para ele, assim vai remover o item no índice especificado. Se o índice não existir, irá gerar um IndexError.
O método remove irá remover um item pelo valor dele, e não pelo índice. Se o valor não existir, irá gerar um ValueError.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> planetas.remove('Terra')
>>> planetas
['Mercúrio', 'Vênus', 'Marte']
>>> planetas.count('Terra')
0
>>> segundo = planetas.pop(1)
>>> planetas
['Mercúrio', 'Marte']
>>> segundo
'Vênus'
```

O método sort irá ordenar a lista em ordem crescente, por padrão.
Se quisermos ordenar de trás para frente, podemos usar o método sort com o argumento nomeado "reverse=True", ou então usar o método reverse.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> planetas.sort()
>>> planetas
['Marte', 'Mercúrio', 'Terra', 'Vênus']
>>> planetas.sort(reverse=True)
>>> planetas
['Vênus', 'Terra', 'Mercúrio', 'Marte']
>>> planetas.sort()
>>> planetas
['Marte', 'Mercúrio', 'Terra', 'Vênus']
>>> planetas.reverse()
>>> planetas
['Vênus', 'Terra', 'Mercúrio', 'Marte']
```

Função len

A função len tem a funcionalidade de retornar o tamanho da string ou da lista.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> len(planetas)
4
>>> nome = 'Gandalf, o Cinzento'
>>> len(nome)
19
```

Índices e Strings

As strings nada mais são do que uma série de caracteres. Logo, também podemos usar algumas operações de índices com elas.
As strings também podem ser acessadas usando índices, mas não podem ter seus valores alterados, pois as strings são um tipo imutável.

```
>>> nome = 'Gandalf, o Cinzento'
>>> nome[5:]
'lf, o Cinzento'
>>> nome[:6]
'Gandal'
>>> parte = nome[3:8]
>>> parte
'dalf,'
```

Se tentarmos alterar apenas um caractere, vamos gerar um TypeError.

```
>>> nome = 'Gandalf, o Cinzento'
>>> nome[0]
'G'
>>> nome[0] = 'F'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Listas e Strings

As strings podem ser divididas em uma lista usando o método split.
Por padrão, a string será separada usando o espaço como padrão, mas podemos alterar isso passando uma string como argumento.

```
>>> nome = 'Gandalf, o Cinzento'
>>> nome.split()
['Gandalf,', 'o', 'Cinzento']
>>> nome.split(',')
['Gandalf', ' o Cinzento']
```

E podemos criar strings a partir de listas com o método join.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> ''.join(planetas)
'MercúrioVênusTerraMarte'
>>> '-'.join(planetas)
'Mercúrio-Vênus-Terra-Marte'
>>> numeros = [1,2,3,4]
>>> '.'.join(numeros)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: sequence item 0: expected str instance, int found
```

A string que passamos junto do join é o que será usado como separados de todos os itens da lista. Se tentarmos passar algo que não seja uma string, vamos gerar um TypeError.

Comando in

O comando in pode ser usado para ver se um determinado item numa lista ou um determinado caractere na string.

```
>>> planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']
>>> 'Terra' in planetas
True
>>> 'Plutão' in planetas
False
>>> nome = 'Gandalf, o Cinzento'
>>> 'andalf' in nome
True
>>> 'Sauron' in nome
False
```

Como ele retorna um booleano, podemos usar isso nas condições de validação do comando if também.

```python
planetas = ['Mercúrio', 'Vênus', 'Terra', 'Marte']

# se o planeta não estiver na lista, insere ele
if 'Júpiter' not in planetas:
    planetas.append('Júpiter')
    print(planetas)
else:
    print('Júpiter já está na lista')
```

Dicas VS Code

Comando: descrição

- ctrl + s -> salva o documento
- ctrl + a -> seleciona todo o documento
- ctrl + d -> seleciona a palavra onde o cursor está posicionado, se continuar apertando, vai selecionando as palavras iguais subsequentes
- shift + alt + seta para baixo -> duplica a linha onde está o cursor para baixo
- shift + alt + seta para cima -> duplica a linha onde está o cursor para cima
- alt + z -> ativa ou desativa a quebra de linha automática (ativando evita de ter que deslocar a tela para direita)

Exercícios para Praticar

1. Crie um algoritmo que verifique se um número é par ou ímpar.
2. Crie um algoritmo que verifique se um número é múltiplo de 3, 5, 7 ou 11.
3. Crie um algoritmo que verifique se um número é múltiplo de 3, 5 e 7.
4. Crie um algoritmo que verifique se um número é múltiplo de 3 e 5.
5. Crie um algoritmo que verifique se um número é múltiplo de 3 e 7.
6. Crie um algoritmo que verifique se um número é múltiplo de 5 e 7.
7. Crie um algoritmo que verifique se um número é múltiplo de 3 ou 5.
8. Crie um algoritmo que verifique se um número é múltiplo de 3 ou 7.
9. Crie um algoritmo que verifique se um número é múltiplo de 5 ou 7.
10. Crie um algoritmo que verifique se um número é múltiplo de 3 e não é múltiplo de 5.
11. Crie um algoritmo que verifique se um número não é múltiplo de 3 e é múltiplo de 5.
12. Crie um algoritmo que verifique se um número é múltiplo de 3 e não é múltiplo de 7.
13. Crie um algoritmo que verifique se um número não é múltiplo de 3 e é múltiplo de 7.
14. Crie um algoritmo que verifique se um número é múltiplo de 5 e não é múltiplo de 7.
15. Crie um algoritmo que verifique se um número é múltiplo de 3 e seja par.
16. Crie um algoritmo que verifique se um número é múltiplo de 3 e não seja par.
17. Crie um algoritmo que verifique se um número é múltiplo de 3 e seja ímpar.
18. Crie um algoritmo que verifique se um número é múltiplo de 3 e não seja ímpar.
19. Crie um algoritmo que verifique se um número é múltiplo de 5 e seja par.
20. Crie um algoritmo que verifique se um número é múltiplo de 5 e não seja par.
21. Crie um algoritmo que verifique se um número é múltiplo de 5 e seja ímpar.
22. Crie um algoritmo que verifique se um número é múltiplo de 5 e não seja ímpar.
23. Crie um algoritmo que verifique se uma palavra possui qualquer vogal.
24. Crie um algoritmo que verifique se uma palavra não possui qualquer vogal.
25. Crie uma lista vazia e adicione os números 10, 20 e 30 utilizando o método append.
26. Crie uma lista com os números 10, 20 e 30. Utilize o método pop para remover o último elemento da lista.
27. Crie uma lista com os números 30, 20 e 10. Utilize o método sort para ordenar a lista em ordem crescente.
28. Crie uma lista com os números 10, 20 e 30. Utilize o método reverse para inverter a ordem dos elementos na lista.
29. Crie uma lista com os nomes "João", "Maria" e "Ana". Utilize a função len para saber quantos nomes há na lista.
30. Crie uma lista com os números 10, 20 e 30. Utilize a condição if para verificar se o número 20 está na lista.
31. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Utilize a condição if para verificar se a palavra "avião" está na lista.
32. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Utilize a condição elif para verificar se a palavra "carro" está na lista e exibir uma mensagem.
33. Crie uma lista com os números 10, 20 e 30. Utilize a condição if para verificar se a soma dos números é maior que 50.
34. Crie uma lista com os números 10, 20 e 30. Utilize a condição else para exibir uma mensagem caso o número 40 não esteja na lista.
35. Crie uma lista com as palavras "vermelho", "azul" e "verde". Utilize o método sort para ordenar a lista em ordem alfabética.
36. Crie uma lista com os números 10, 20 e 30. Utilize o método pop para remover o elemento com índice 1.
37. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Utilize o método reverse para inverter a ordem das palavras na lista.
38. Crie uma lista vazia e adicione o nome "João" utilizando o método append.
39. Crie uma lista com os números 10, 20 e 30. Utilize o método pop para remover o primeiro elemento da lista.
40. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Utilize a condição if para verificar se a palavra "carro" está na lista e, se estiver, utilize o método pop para removê-la.
41. Crie uma lista com os números 10, 20 e 30. Utilize a função len para saber quantos elementos há na lista.

42. Crie uma lista com os números 10, 20 e 30. Utilize a condição elif para verificar se o número 40 está na lista e exibir uma mensagem.
43. Crie uma lista com as palavras "vermelho", "azul" e "verde". Utilize o método pop para remover o último elemento da lista.
44. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Utilize o método sort para ordenar a lista em ordem reversa.
45. Crie uma lista com os números 10, 20 e 30. Acesse o primeiro elemento da lista utilizando o índice.
46. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Acesse o terceiro elemento da lista utilizando o índice.
47. Crie uma lista com os números 10, 20 e 30. Divida a lista em duas partes iguais utilizando índices.
48. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Divida a lista em duas partes iguais utilizando índices.
49. Crie uma lista com os números 10, 20 e 30. Acesse o último elemento da lista utilizando o índice.
50. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Acesse o primeiro elemento da lista utilizando o índice.
51. Crie uma lista com os números 10, 20 e 30. Divida a lista em três partes iguais utilizando índices.
52. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Divida a lista em quatro partes iguais utilizando índices.
53. Crie uma lista com os números 10, 20 e 30. Acesse o segundo elemento da lista utilizando o índice.
54. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Acesse o último elemento da lista utilizando o índice.
55. Crie uma lista com os números 10, 20 e 30. Divida a lista em duas partes, a primeira contendo o primeiro e segundo elementos e a segunda contendo o terceiro elemento.
56. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Divida a lista em duas partes, a primeira contendo o primeiro e segundo elementos e a segunda contendo o terceiro e quarto elementos.
57. Crie uma lista com os números 10, 20 e 30. Acesse o último elemento da lista utilizando índice negativo.
58. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Acesse o primeiro elemento da lista utilizando índice negativo.
59. Crie uma lista com os números 10, 20 e 30. Divida a lista em três partes, a primeira contendo o primeiro elemento, a segunda contendo o segundo elemento e a terceira contendo o terceiro elemento.
60. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Divida a lista em quatro partes, a primeira contendo o primeiro elemento, a segunda contendo o segundo elemento, a terceira contendo o terceiro elemento e a quarta contendo o quarto elemento.
61. Crie uma lista com os números 10, 20 e 30. Acesse o segundo elemento da lista utilizando índice negativo.
62. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Acesse o segundo elemento da lista utilizando índice negativo.
63. Crie uma lista com os números 10, 20 e 30. Divida a lista em duas partes, a primeira contendo o primeiro elemento e a segunda contendo o segundo e terceiro elementos.
64. Crie uma lista com as palavras "casa", "carro", "bicicleta" e "moto". Divida a lista em duas partes, a primeira contendo o primeiro e terceiro elementos e a segunda contendo o segundo e quarto elementos.
65. Crie uma string com a frase "O rato roeu a roupa do rei de Roma". Acesse o primeiro caractere da string utilizando o índice.
66. Crie uma string com a palavra "Python". Acesse o último caractere da string utilizando o índice.
67. Crie uma string com a frase "Amanhã é sexta-feira". Acesse o quarto caractere da string utilizando o índice.
68. Crie uma string com a palavra "Futebol". Acesse o segundo caractere da string utilizando o índice.
69. Crie uma string com a frase "O tempo está bom hoje". Utilize a função len para contar quantos caracteres tem na string.
70. Crie uma string com a palavra "Abacaxi". Utilize a função len para contar quantos caracteres tem na string.
71. Crie uma string com a frase "O gato mia quando está com fome". Acesse o último caractere da string utilizando índice negativo.
72. Crie uma string com a palavra "Pneumático". Acesse o terceiro caractere da string utilizando índice negativo.

73. Crie uma string com a frase "Amanhã é dia de ir ao cinema". Divida a string em duas partes iguais utilizando índices.
74. Crie uma string com a palavra "Paralelepípedo". Divida a string em duas partes iguais utilizando índices.
75. Crie uma string com a frase "Os pássaros voam no céu azul". Acesse o primeiro caractere da palavra "céu" utilizando o índice.
76. Crie uma string com a palavra "Hipopótamo". Acesse o último caractere da palavra utilizando o índice negativo.
77. Crie uma string com a frase "O cachorro late quando vê um desconhecido". Divida a string em três partes iguais utilizando índices.
78. Crie uma string com a palavra "Estatística". Acesse o segundo caractere da palavra utilizando o índice negativo.
79. Crie uma string com a frase "Eu gosto de sorvete de chocolate". Utilize a função len para contar quantas palavras tem na string.
80. Crie uma string com a palavra "Azeitona". Utilize a função len para contar quantas vogais tem na string.
81. Crie uma string com a frase "O pássaro canta na árvore". Acesse o último caractere da palavra "árvore" utilizando índice negativo.
82. Crie uma string com a palavra "Papagaio". Divida a string em duas partes, a primeira contendo os dois primeiros caracteres e a segunda contendo os quatro últimos caracteres.
83. Crie uma string com a frase "O sol brilha no céu". Acesse o primeiro caractere da palavra "brilha" utilizando o índice.
84. Crie uma string com a palavra "Pneumoultramicroscopicossilicovulcanoconiótico". Utilize a função len para contar quantas letras tem na palavra.
85. Dado um nome completo em uma string separada por espaço, verifique se a pessoa tem um sobrenome. Use o método split e uma condição if.
86. Dada uma lista de palavras, junte-as em uma única string separada por vírgulas. Use o método join.
87. Dada uma string com a frase "O cachorro correu atrás do gato", separe cada palavra em uma lista. Use o método split.
88. Dado um número inteiro, verifique se é um número par ou ímpar. Converta-o para uma string e verifique o último caractere. Use uma condição if.
89. Dada uma lista com as cores "azul", "verde", "amarelo" e "vermelho", junte-as em uma única string separada por vírgulas e adicione a palavra "e" antes da última cor. Use o método join.
90. Dada uma string com a frase "Eu amo programar em Python", separe cada palavra em uma lista e verifique se a palavra "Python" está na lista. Use o método split e uma condição if.
91. Dado um número real, verifique se é positivo ou negativo. Converta-o para uma string e verifique o primeiro caractere. Use uma condição if.
92. Dada uma lista com os números 2, 4, 6 e 8, junte-os em uma única string separada por vírgulas e adicione a palavra "e" antes do último número. Use o método join.
93. Dada uma string com a frase "A minha casa é azul", separe cada palavra em uma lista e verifique se a palavra "vermelho" está na lista. Use o método split e uma condição if.
94. Dado um número inteiro, verifique se é maior ou menor que 10. Converta-o para uma string e verifique o primeiro caractere. Use uma condição if.
95. Dada uma lista com as frutas "maçã", "banana", "laranja" e "abacaxi", junte-as em uma única string separada por vírgulas e adicione a palavra "e" antes da última fruta. Use o método join.
96. Dada uma string com a frase "Eu gosto de andar de bicicleta", separe cada palavra em uma lista e verifique se a palavra "bicicleta" está na lista. Use o método split e uma condição if.
97. Dado um número real, verifique se é um número inteiro ou um número decimal. Converta-o para uma string e verifique se contém o caractere ".". Use uma condição if.
98. Dada uma lista com os números 3, 6, 9 e 12, junte-os em uma única string separada por vírgulas e adicione a palavra "e" antes do último número. Use o método join.
99. Dada uma string com a frase "O meu carro é vermelho", separe cada palavra em uma lista e verifique se a palavra "verde" está na lista. Use o método split e uma condição if.
100. Dado um número inteiro, verifique se é positivo, negativo ou zero. Converta-o para uma string e verifique o primeiro caractere. Use uma condição if.

101. Dada uma lista com as cores "branco", "preto", "cinza" e "marrom", junte-as em uma única string separada por vírgulas e adicione a palavra "ou" antes da última cor. Use o método join.
102. Dada uma string com a frase "Eu moro em São Paulo", separe cada palavra em uma lista e verifique se a palavra "Rio de Janeiro" está na lista. Use o método split e uma condição if.
103. Dado um número real, verifique se é um número positivo, negativo ou zero. Converta-o para uma string e verifique o primeiro caractere. Use uma condição if.
104. Dada uma lista com as cidades "São Paulo", "Rio de Janeiro", "Belo Horizonte" e "Curitiba", junte-as em uma única string separada por vírgulas e adicione a palavra "e" antes da última cidade. Use o método join.