Teste de Software para Web

Revisão de Lógica de Programação

MSc. Jonathas Silva dos Santos

Roteiro

Expressões Relacionais

Expressões Lógicas

Comandos Condicionais

Expressões Relacionais

Tipo bool

- Já vimos que o tipo **bool** é utilizado para representar os valores booleanos verdadeiro (**True**) e falso (**False**).

```
1 a = True
2 b = False
```

- O uso mais comum dessas variáveis é na verificação de expressões relacionais e lógicas.

Expressões Relacionais

- Expressões relacionais são aquelas que realizam uma comparação entre duas expressões e retornam:
  - **True**, se o resultado for verdadeiro.
  - **False**, se o resultado for falso.
- Os operadores relacionais são:

  `==` igualdade.
  `!=` diferente.
  `>` maior que.
  `<` menor que.
  `>=` maior ou igual que.
  `<=` menor ou igual que.
- Nos próximos exemplos, considere que foram feitas as seguintes atribuições:

```
1 a = 20
2 b = 21
```

Expressões Relacionais

- <expressão> `==` <expressão>: retorna verdadeiro quando as expressões forem iguais.

```
a == (10 * 2) # a = 20
# True
b == (10 * 2) # b = 21
# False
```

- <expressão> `!=` <expressão>: retorna verdadeiro quando as expressões forem diferentes.

```
a != (10 * 2) # a = 20
# False
b != (10 * 2) # b = 21
# True
```

Expressões Relacionais

- <expressão> **>** <expressão>: retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita.

```
# a = 20 e b = 21
a > b
# False
```

- <expressão> **<** <expressão>: retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita.

```
# a = 20 e b = 21
a < b
# True
```

Expressões Relacionais

- <expressão> **>=** <expressão>: retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.

```
# a = 20 e b = 21
a >= b
# False
```

- <expressão> **<=** <expressão>: retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.

```
# a = 20 e b = 21
a <= b
# True
```

Expressões Relacionais com Strings

- Ordem considerada para os caracteres do alfabeto:
  - *ABC...XYZabc...xyz*

```
"a" > "b"
# False
"a" == "a"
# True
"a" == "A"
# False
"Z" < "a"
# True
"z" < "a"
# False

```

Expressões Relacionais com Strings

- Ordem considerada para os caracteres do alfabeto:
  - *ABC...XYZabc...xyz*

```
"azzzz" < "zaaaa"
# True
"azzzz" < "Zaaaa"
# False
"3" == 3
# False
3 > "4"
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# TypeError: '>' not supported between instances of 'int'
    and 'str'
```

Exemplo

O que será impresso pelo código a seguir?

```
print((3 * 4) / 2 == (2 * 3))
# True
print((4 / 3) <= 1.33)
# False
```

Expressões Lógicas

Expressões Lógicas

- Expressões lógicas são aquelas que realizam uma operação lógica e retornam verdadeiro ou falso (como as expressões relacionais).
- Os operadores lógicos são:
  - `and`: operador E.
  - `or`: operador OU.
  - `not`: operador NÃO.

Operador Lógico and

- <expressão1> **and** <expressão2>: retorna verdadeiro quando ambas as expressões são verdadeiras.
- Sua tabela verdade é:

<expressão1>	<expressão2>	resultado
True	True	True
True	False	False
False	True	False
False	False	False

```
a = 5
b = 10
print((a > 0) and (b == 0))
# False
print((a > 0) and (b != 0))
# True
```

Operador Lógico or

- <expressão1> **or** <expressão2>: retorna verdadeiro quando pelo menos uma das expressões é verdadeira.
- Sua tabela verdade é:

<expressão1>	<expressão2>	resultado
True	True	True
True	False	True
False	True	True
False	False	False

```
1 a = 5
2 b = 10
3 print((a > 0) or (b == 0))
4 # True
5 print((a != 5) or (b == 0))
6 # False
```

Operador Lógico not

- `not` <expressão>: retorna verdadeiro quando a expressão é falsa e vice-versa.
- Sua tabela verdade é:

<expressão>	resultado
True	False
False	True

```
1 a = 5
2 b = 10
3 print(not(a < b))
4 # False
5 print(not(a == b))
6 # True
```

Expressões Equivalentes

- `not(a == b)` é equivalente a `(a != b)`
- `not(a > b)` é equivalente a `(a <= b)`
- `not(a < b)` é equivalente a `(a >= b)`

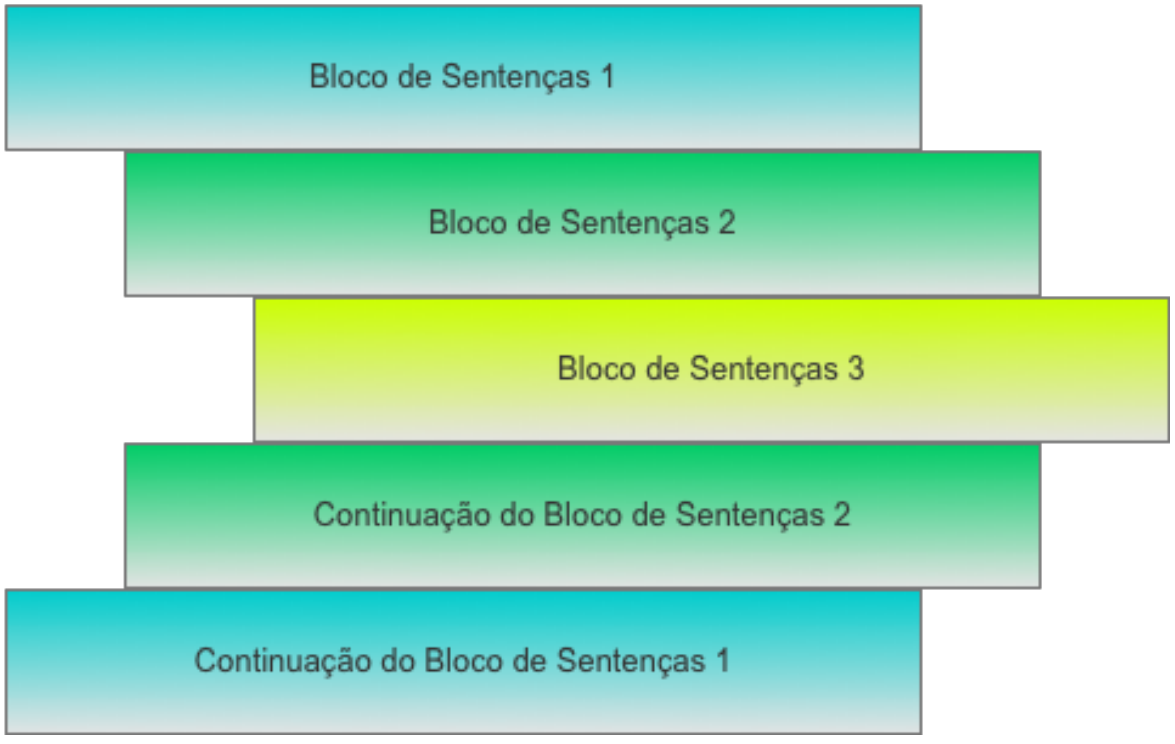
Exemplo

O que será impresso pelo código a seguir?

```
1  a = True
2  b = False
3  print(not(a or b))
4  # False
5  print(not(a and b))
6  # True
7  print(not(a) and not(b))
8  # False
9  print(not(a) or not(b))
10 # True
```

Comandos Condicionais

Blocos



- Um bloco é um conjunto de comandos agrupados.
- Os programas Python são estruturados através de indentação, ou seja, os blocos são definidos pelo seu espaçamento (*tabs*) em relação ao início da linha.

Comandos Condicionais

- O principal comando condicional é o `if`:

```
if <condição>:
# bloco a ser executado se a condição for verdadeira
  <comando1>
  <comando2>
  ...
  <comandoY>
```

- O bloco de comandos é executado somente se a condição (expressão relacional, expressão lógica ou variável booleana) for verdadeira.
- Na estrutura do comando `if` sempre há um “:” após a condição.

Comandos Condicionais

- O programa a seguir verifica se um número inteiro é ímpar.

```
1 a = int(input("Digite um número inteiro: "))
2 impar = ((a % 2) == 1)
3 if impar:
4   print("Número ímpar")
5
6 print("Fim do programa")
```

Comandos Condicionais

- O programa a seguir verifica se um número inteiro é ímpar.

```
1 a = int(input("Digite um número inteiro: "))
2
3 if (a % 2) == 1:
4   print("Número ímpar")
5
6 print("Fim do programa")
```

Comandos Condicionais

- O programa a seguir verifica se um número inteiro é par ou ímpar.

```
1 a = int(input("Digite um número inteiro: "))
2
3 if (a % 2) == 0:
4   print("Número par")
5 if (a % 2) == 1:
6   print("Número ímpar")
7
8 print("Fim do programa")
```

Comandos Condicionais

- Uma variação do comando `if` é o `if/else`:

```
if <condição>:
  # bloco a ser executado se a condição for verdadeira
  <comando>
  ...
  <comando>
else:
  # bloco a ser executado se a condição for falsa
  <comando>
  ...
  <comando>
```

Comandos Condicionais

- O programa a seguir verifica se um número inteiro é par ou ímpar.

```
1 a = int(input("Digite um número inteiro: "))
2
3 if (a % 2) == 0:
4   print("Número par")
5 else:
6   print("Número ímpar")
7
8 print("Fim do programa")
```

Comandos Condicionais

- O programa a seguir determina o maior entre dois números.

```
a = float(input("Digite o primeiro número: "))
b = float(input("Digite o segundo número: "))

if a > b:
  print("O maior número é", a)
else:
  print("O maior número é", b)
```

Comandos Condicionais

- O programa a seguir compara dois números.

```
1  a = float(input("Digite o primeiro número: "))
2  b = float(input("Digite o segundo número: "))
3
4  if a == b:
5    print("Os dois números são iguais")
6  else:
7    if a > b:
8      print("O maior número é o primeiro")
9    else:
10     print("O maior número é o segundo")
```

Comandos Condicionais

- Podemos simplificar o código anterior utilizando **`elif`**.

```
1  a = float(input("Digite o primeiro número: "))
2  b = float(input("Digite o segundo número: "))
3
4  if a == b:
5    print("Os dois números são iguais")
6  elif a > b:
7    print("O maior número é o primeiro")
8  else:
9    print("O maior número é o segundo")
10
```

Comandos Condicionais

- O comando **`elif`** é utilizado quando queremos fazer o teste de várias alternativas.

```
ra = input("Entre com o RA de um aluno: ")
if ra == "155446":
  print("Gabriel Siqueira")
elif ra == "192804":
  print("Alexsandro Alexandrino")
elif ra == "209823":
  print("Ana Paula Dantas")
elif ra == "188948":
  print("Klairton Brito")
# ...
elif ra == "999999":
  print("...")
else:
  print("Aluno não encontrado")
```

Exemplo

```
1  a = int(input())
2
3  if a > 3:
4    if a < 7:
5      print("a")
6  else:
7    if a > -10:
8      print("b")
9    else:
10     print("c")
```

- No código acima, o que será impresso...
  - ... quando `a = 5`? “a”.
  - ... quando `a = 10`? Nada.
  - ... quando `a = -5`? “b”.
  - ... quando `a = -15`? “c”.

Exemplo

```
1  a = int(input())
2
3  if a > 3:
4    if a < 7:
5      print("a")
6    else:
7      if a > -10:
8        print("b")
9      else:
10       print("c")
```

- No código acima, o que será impresso...
  - ... quando `a = 5`? “a”.
  - ... quando `a = 10`? “b”.
  - ... quando `a = -5`? Nada.
  - ... quando `a = -15`? Nada.

Exemplo

```
1  a = int(input())
2
3  if a > 3:
4    if a < 7:
5      print("a")
6    else:
7      if a > -10:
8        print("b")
9  else:
10   print("c")
```

- No código acima, o que será impresso...
  - ... quando `a = 5`? “a”.
  - ... quando `a = 10`? “b”.
  - ... quando `a = -5`? “c”.
  - ... quando `a = -15`? “c”.

Exercícios

1. Escreva um programa que, dados três números inteiros, imprima o menor deles.
2. Escreva um programa que, dados três números inteiros, imprima os números em ordem crescente.
3. Escreva um programa que, dadas duas datas, determine qual delas ocorreu cronologicamente primeiro. Para cada uma das duas datas, leia três números referentes ao dia, mês e ano, respectivamente.

Exercício 1 - Resposta

- Escreva um programa que, dados três números inteiros, imprima o menor deles.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))

if (a <= b) and (a <= c):
  print(a)
if (b <= a) and (b <= c):
  print(b)
if (c <= a) and (c <= b):
  print(c)
```

Exercício 1 - Resposta

- Este programa tem um comportamento indesejado quando o menor número não é único. Como corrigi-lo?

```
1  a = int(input("Digite o primeiro número: "))
2  b = int(input("Digite o segundo número: "))
3  c = int(input("Digite o terceiro número: "))
4
5  if (a <= b) and (a <= c):
6    print(a)
7  if (b <= a) and (b <= c):
8    print(b)
9  if (c <= a) and (c <= b):
10   print(c)
```

Exercício 1 - Resposta

- Escreva um programa que, dados três números inteiros, imprima o menor deles.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))

if (a <= b) and (a <= c):
  print(a)
elif (b <= c):
  print(b)
else:
  print(c)
```

Exercício 2 - Resposta

- Escreva um programa que, dados três números inteiros, imprima os números em ordem crescente.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))
if (a <= b) and (b <= c):
  print(a, b, c)
elif (a <= c) and (c <= b):
  print(a, c, b)
elif (b <= a) and (a <= c):
  print(b, a, c)
elif (b <= c) and (c <= a):
  print(b, c, a)
elif (c <= a) and (a <= b):
  print(c, a, b)
elif (c <= b) and (b <= a):
  print(c, b, a)
```

Exercício 2 - Resposta

- Escreva um programa que, dados três números inteiros, imprima os números em ordem crescente.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))
if (a <= b) and (b <= c):
  print(a, b, c)
elif (a <= c) and (c <= b):
  print(a, c, b)
elif (b <= a) and (a <= c):
  print(b, a, c)
elif (b <= c) and (c <= a):
  print(b, c, a)
elif (c <= a) and (a <= b):
  print(c, a, b)
else:
  print(c, b, a)
```

Exercício 2 - Resposta

- Escreva um programa que, dados três números inteiros, imprima os números em ordem crescente.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))
if (a <= b <= c):
  print(a, b, c)
elif (a <= c <= b):
  print(a, c, b)
elif (b <= a <= c):
  print(b, a, c)
elif (b <= c <= a):
  print(b, c, a)
elif (c <= a <= b):
  print(c, a, b)
else:
  print(c, b, a)
```

Exercício 2 - Resposta

- Escreva um programa que, dados três números inteiros, imprima os números em ordem crescente.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))

if (a <= b) and (a <= c): # O menor é o primeiro (a)
  if (b <= c):
    print(a, b, c)
  else:
    print(a, c, b)
elif (b <= c):            # O menor é o segundo (b)
  if (a <= c):
    print(b, a, c)
  else:
    print(b, c, a)
# ...
```

Exercício 2 - Resposta (Continuação)

- Escreva um programa que, dados três números inteiros, imprima os números em ordem crescente.

```
# ...
else:                          # O menor é o terceiro (c)
  if (a <= b):
    print(c, a, b)
  else:
    print(c, b, a)
```

Funções `min` e `max`

- Python possui as funções **`min`** (mínimo) e **`max`** (máximo).
- A função **`min`** retorna o menor valor dentre todos os valores passados como argumento.

```
a = 5
b = 10
print(min(100, a, 7, b))
# 5
```

- A função **`max`** retorna o maior valor dentre todos os valores passados como argumento.

```
print(max(100, a, 7, b))
# 100
```

- Refaça os dois exercícios anteriores sem utilizar comandos condicionais. Dica: use as funções **`min`** e **`max`**.

Exercício 3 - Resposta

- Escreva um programa que, dadas duas datas, determine qual delas ocorreu cronologicamente primeiro. Para cada uma das duas datas, leia três números referentes ao dia, mês e ano, respectivamente.

```
1  dia1 = int(input("Digite o dia da primeira data: "))
2  mes1 = int(input("Digite o mês da primeira data: "))
3  ano1 = int(input("Digite o ano da primeira data: "))
4
5  dia2 = int(input("Digite o dia da segunda data: "))
6  mes2 = int(input("Digite o mês da segunda data: "))
7  ano2 = int(input("Digite o ano da segunda data: "))
8
9
10
11
12
13 # ...
```

Exercício 3 - Resposta (Continuação)

- Escreva um programa que, dadas duas datas, determine qual delas ocorreu cronologicamente primeiro. Para cada um das duas datas, leia três números referentes ao dia, mês e ano, respectivamente.

```
# ...
if ano1 < ano2:
  print(dia1, mes1, ano1, sep="/")
elif ano2 < ano1:
  print(dia2, mes2, ano2, sep="/")
elif mes1 < mes2:
  print(dia1, mes1, ano1, sep="/")
elif mes2 < mes1:
  print(dia2, mes2, ano2, sep="/")
elif dia1 < dia2:
  print(dia1, mes1, ano1, sep="/")
else:
  print(dia2, mes2, ano2, sep="/")
```

Exercícios

4. Escreva um programa que calcule as raízes de uma equação de segundo grau. O seu programa deve receber três números a, b e c, sendo que a equação é definida como $ax^2 + bx + c = 0$. O seu programa também deve tratar o caso em que $a = 0$.
5. Escreva um programa que simula o jogo conhecido como "Pedra, Papel e Tesoura" de um jogador A contra um jogador B. O programa deve ler a escolha do jogador A e a escolha do jogador B. Por fim, o programa deve indicar quem foi o vencedor.

Exercício 4 - Equação do Segundo Grau

- Equação do 2º grau:

$$ax^2 + bx + c = 0$$

- Fórmula de Bhaskara:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

$$\Delta = b^2 - 4ac$$

Exercício 4 - Resposta

- Escreva um programa que calcule as raízes de uma equação de segundo grau. O seu programa deve receber três números a, b e c, sendo que a equação é definida como $ax^2 + bx + c = 0$. O seu programa também deve tratar o caso em que $a = 0$.

```
1  a = float(input("Digite o coeficiente a: "))
2  b = float(input("Digite o coeficiente b: "))
3  c = float(input("Digite o coeficiente c: "))
4
5
6  if a == 0: # equação do primeiro grau
7    if b == 0:
8      print("Não existe raíz.")
9    else:
10     raiz = (-c / b)
11     print("A raiz é:", raiz)
12 # ...
```

Exercício 4 - Resposta (Continuação)

- Escreva um programa que calcule as raízes de uma equação de segundo grau. O seu programa deve receber três números a, b e c, sendo que a equação é definida como $ax^2 + bx + c = 0$. O seu programa também deve tratar o caso em que $a = 0$.

```
# ...
else: # equação do segundo grau
  delta = (b ** 2) - (4 * a * c)
  if delta < 0:
    print("Não existem raízes reais.")
  elif delta != 0:
    raiz1 = (-b + delta ** (1 / 2)) / (2 * a)
    raiz2 = (-b - delta ** (1 / 2)) / (2 * a)
    print("As raízes são:", raiz1, "e", raiz2)
  else:
    raiz = -b / (2 * a)
    print("A raiz é:", raiz)
```

Exercício 5 - Resposta

- Escreva um programa que simula o jogo conhecido como "Pedra, Papel e Tesoura" de um jogador A contra um jogador B. O programa deve ler a escolha do jogador A e a escolha do jogador B. Por fim, o programa deve indicar quem foi o vencedor.

```
1  jogadorA = input("Digite a primeira escolha: ")
2  jogadorB = input("Digite a segunda escolha: ")
3
4  if jogadorA == "pedra":
5    if jogadorB == "pedra":
6      print("Empate")
7    elif jogadorB == "tesoura":
8      print("O jogador A ganhou")
9    else:
10     print("O jogador B ganhou")
11 # ...
```

Exercício 5 - Resposta (Continuação)

- Escreva um programa que simula o jogo conhecido como "Pedra, Papel e Tesoura" de um jogador A contra um jogador B. O programa deve ler a escolha do jogador A e a escolha do jogador B. Por fim, o programa deve indicar quem foi o vencedor.

```
# ...
elif jogadorA == "tesoura":
  if jogadorB == "pedra":
    print("O jogador B ganhou")
  elif jogadorB == "tesoura":
    print("Empate")
  else:
    print("O jogador A ganhou")
# ...
```

Exercício 5 - Resposta (Continuação)

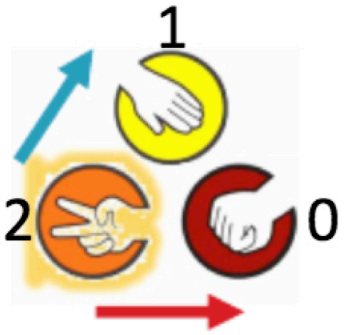
- Escreva um programa que simula o jogo conhecido como "Pedra, Papel e Tesoura" de um jogador A contra um jogador B. O programa deve ler a escolha do jogador A e a escolha do jogador B. Por fim, o programa deve indicar quem foi o vencedor.

```
# ...
else: # jogadorA == "papel"
  if jogadorB == "pedra":
    print("O jogador A ganhou")
  elif jogadorB == "tesoura":
    print("O jogador B ganhou")
  else:
    print("Empate")
```

Exercício 5 - Nova Versão

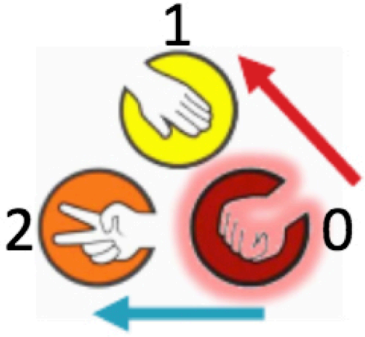
- Associar objetos a números é uma forma de abstração. No código a seguir usamos as seguintes associações:
  - `pedra = 0`
  - `papel = 1`
  - `tesoura = 2`
- O resultado da expressão `(jogadorA - jogadorB) % 3` indica, de forma única, o vencedor da partida.
- Complete o código analisando o resultado da expressão anterior.

Exercício 5 - Nova Versão




```
(2 - 1) % 3 = 1
(2 - 0) % 3 = 2
```


```
(1 - 0) % 3 = 1
(1 - 2) % 3 = 2
```




```
(0 - 2) % 3 = 1
(0 - 1) % 3 = 2
```

Exercício 5 - Nova Versão

```python
print("Pedra = 0")
print("Papel = 1")
print("Tesoura = 2")

jogadorA = int(input("Digite a primeira escolha: "))
jogadorB = int(input("Digite a segunda escolha: "))

resultado = (jogadorA - jogadorB) % 3

if resultado == 1:
  print("O jogador A ganhou")
elif resultado == 2:
  print("O jogador B ganhou")
else:
  print("Empate")
```

Variável *flag*

- Podemos usar uma variável para armazenar um estado do programa.
- Por exemplo, podemos criar uma variável para indicar se um sistema está funcionando corretamente (ou se apresentou alguma falha).
- Normalmente inicializamos esta variável com um valor padrão (por exemplo, `True`) e atualizamos a variável caso uma mudança de estado ocorra (trocando o valor, por exemplo, para `False`).
- Este tipo de variável, que serve para sinalizar uma situação específica, é chamada de *flag*.
- Uma variável *flag* pode simplificar significativamente a escrita, manutenção e o entendimento de um programa.

Exemplo sem *flag*

```
...


if <condição1>:
  print("Falha do tipo 1")

if <condição2>:
  print("Falha do tipo 2")

if <condição3>:
  print("Falha do tipo 3")


...
```

Exemplo sem *flag*

```
...


if <condição4>:
  print("Falha do tipo 4")

...

if <condição100>:
  print("Falha do tipo 100")


if not(<condição1>) and not(<condição2>) ... not(<condição100>):
  print("Sistema funcionando normalmente")
```

Exemplo com *flag*

```
1  OK = True
2
3
4  if <condição1>:
5    print("Falha do tipo 1")
6    OK = False
7  if <condição2>:
8    print("Falha do tipo 2")
9    OK = False
10 if <condição3>:
11   print("Falha do tipo 3")
12   OK = False
13
14 ...
```

Exemplo com *flag*

```
...


if <condição4>:
  print("Falha do tipo 4")
  OK = False
...

if <condição100>:
  print("Falha do tipo 100")
  OK = False

if OK:
  print("Sistema funcionando normalmente")
```