Notas do curso básico de programação em Python

Encontro 3 - Estruturas condicionais

Prof. Louis Augusto
louis.augusto@ifsc.edu.br

Instituto Federal de Santa Catarina
Campus São José

Índice

Sumário

Operadores relacionais

Python suporta uma boa gama de operadores lógicos que permitem comparações matemáticas entre variáveis, usando estruturas de decisão if e while.

é igual a	$a == b$
é diferente de	$a != b$
menor que	a<b
menor ou igual que	a<=b
maior que	a>b
maior ou igual que	a>=b

Estes operadores relacionais são usados para que o programa possa tomar decisões entre vários casos possíveis.
De forma bem simples, se fizermos `print(3>5)` a impressão em tela será a palavra `False`.

Vamos estudar os casos em particular.

Operadores relacionais

Python suporta uma boa gama de operadores lógicos que permitem comparações matemáticas entre variáveis, usando estruturas de decisão if e while.

é igual a	$a == b$
é diferente de	$a != b$
menor que	a<b
menor ou igual que	a<=b
maior que	a>b
maior ou igual que	a>=b

Estes operadores relacionais são usados para que o programa possa tomar decisões entre vários casos possíveis.
De forma bem simples, se fizermos `print(3>5)` a impressão em tela será a palavra `False`.

Vamos estudar os casos em particular.

Operadores relacionais

Python suporta uma boa gama de operadores lógicos que permitem comparações matemáticas entre variáveis, usando estruturas de decisão if e while.

é igual a	$a == b$
é diferente de	$a != b$
menor que	a<b
menor ou igual que	a<=b
maior que	a>b
maior ou igual que	a>=b

Estes operadores relacionais são usados para que o programa possa tomar decisões entre vários casos possíveis.

De forma bem simples, se fizermos `print(3>5)` a impressão em tela será a palavra `False`.

Vamos estudar os casos em particular.

Operadores relacionais

Python suporta uma boa gama de operadores lógicos que permitem comparações matemáticas entre variáveis, usando estruturas de decisão if e while.

é igual a	$a == b$
é diferente de	$a != b$
menor que	a<b
menor ou igual que	a<=b
maior que	a>b
maior ou igual que	a>=b

Estes operadores relacionais são usados para que o programa possa tomar decisões entre vários casos possíveis.
De forma bem simples, se fizermos `print(3>5)` a impressão em tela será a palavra `False`.

Vamos estudar os casos em particular.

Operadores relacionais

Python suporta uma boa gama de operadores lógicos que permitem comparações matemáticas entre variáveis, usando estruturas de decisão if e while.

é igual a	$a == b$
é diferente de	$a != b$
menor que	a<b
menor ou igual que	a<=b
maior que	a>b
maior ou igual que	a>=b

Estes operadores relacionais são usados para que o programa possa tomar decisões entre vários casos possíveis.
De forma bem simples, se fizermos `print(3>5)` a impressão em tela será a palavra `False`.

Vamos estudar os casos em particular.

Operadores == e ! =

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador ! = "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- a 5 != 7 dará True
- a 5 != 5 dará False

Operadores == e ! =

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador ! = "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- a 5 != 7 dará True
- a 5 != 5 dará False

Operadores == e !=

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador != "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- 5 != 7 dará True
- 5 != 5 dará False

Operadores == e !=

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador != "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- 5 != 7 dará True
- 5 != 5 dará False

Operadores == e ! =

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador ! = "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- 5 != 7 dará True
- 5 == 5 dará False

Operadores == e ! =

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador ! = "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- 5 != 7 dará True
- 5 == 5 dará False

Operadores == e !=

Operador == "igual a"

Retorna True se o primeiro operando é igual ao segundo operando

- 5 == 7 dará False
- 7 == 7 dará True

Operador != "diferente de"

Retorna True se o primeiro operando é diferente do segundo

- 5 != 7 dará True
- 5 == 5 dará False

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$: *será False*
- $7 > 5$: *será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$: *será True*
- $7 > 5$: *será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

5 <= 7 será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

5 <= 7 será True.

Operadores $>$ e $<$

Para construir condições mais complexas, precisaremos aprender outros operadores, que são os operadores relacionais e operadores lógicos.

Operação ($>$ Maior que)

- *Retorna True se o primeiro operando é maior que o segundo operando*
- $5 > 7$*: será False*
- $7 > 5$*: será True*

Operação ($<$ Menor que)

- *Retorna True se o primeiro operando é menor que o segundo operando*
- $5 > 7$*: será True*
- $7 > 5$*: será False*

Aos operadores $<$ e $>$ pode ser adicionado $=$ para serem usados como $\geq$ e $\leq$

$5 <= 7$ será True.

Sumário

Precedência de operadores

Assim como na matemática, a precedência dos operadores é importante pois indica a ordem em que eles serão avaliados em uma expressão que inclua vários operadores. A tabela abaixo indica a ordem em que os operadores serão avaliados (da maior precedência para a menor precedência).

Operador	Descrição
**	Exponenciação
*, /, //, %	Multiplicação, Divisão, Quociente, Resto
+, -	Soma, Subtração
<, <=, >, >=, !=, ==	Comparações
not	Não
and	E
or	Ou

Precedência de operadores

Assim como na matemática, a precedência dos operadores é importante pois indica a ordem em que eles serão avaliados em uma expressão que inclua vários operadores. A tabela abaixo indica a ordem em que os operadores serão avaliados (da maior precedência para a menor precedência).

Operador	Descrição
**	Exponenciação
*, /, //, %	Multiplicação, Divisão, Quociente, Resto
+, -	Soma, Subtração
<, <=, >, >=, !=, ==	Comparações
not	Não
and	E
or	Ou

Sumário

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Operação (Estruturas de decisão)

Estruturas de decisão permitem ao programador definir qual sequência de instruções devem ser executadas de acordo com determinada condição.

Existem três estruturas de decisão em Python que podem ser utilizadas:

1. Estrutura `if-elif-else`
2. Estrutura `while`
3. Estrutura `try-except`

A estrutura `if-elif-else` é de tratamento mais direto e será avaliada neste encontro. A estrutura `while` será estudada em conjunto com os laços e a estrutura `try-except`, por ser mais complexa, e utilizada para tratamento de erros que quebram o programa, será estudada à parte.

Sumário

Introdução

A estrutura `if-elif-else` é a mais usada para realizar desvios de código. A palavra `elif` é a junção das palavras else if, e é utilizada quando há necessidade de mais de um desvio. A instrução `if` é obrigatória, e as demais são opcionais no código.

```
if condição:
    Instruções a serem executadas se a condição retornar VERDADE.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
else condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
```

Lembrando que para cada if, elif ou else o programador poderá definir um bloco de instruções, e este bloco é reconhecido por identação em Python.

Introdução

A estrutura `if-elif-else` é a mais usada para realizar desvios de código. A palavra `elif` é a junção das palavras else if, e é utilizada quando há necessidade de mais de um desvio. A instrução `if` é obrigatória, e as demais são opcionais no código.

```
if condição:
    Instruções a serem executadas se a condição retornar VERDADE.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
else condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
```

Lembrando que para cada if, elif ou else o programador poderá definir um bloco de instruções, e este bloco é reconhecido por identação em Python.

Introdução

A estrutura `if-elif-else` é a mais usada para realizar desvios de código. A palavra `elif` é a junção das palavras else if, e é utilizada quando há necessidade de mais de um desvio. A instrução `if` é obrigatória, e as demais são opcionais no código.

if condição:
    Instruções a serem executadas se a condição retornar VERDADE.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
else condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.

Lembrando que para cada if, elif ou else o programador poderá definir um bloco de instruções, e este bloco é reconhecido por identação em Python.

Introdução

A estrutura `if-elif-else` é a mais usada para realizar desvios de código. A palavra `elif` é a junção das palavras else if, e é utilizada quando há necessidade de mais de um desvio. A instrução `if` é obrigatória, e as demais são opcionais no código.

if condição:
    Instruções a serem executadas se a condição retornar VERDADE.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
elif condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.
else condição: (opcional)
Instruções a serem executadas se a condição retornar FALSA.

Lembrando que para cada if, elif ou else o programador poderá definir um bloco de instruções, e este bloco é reconhecido por identação em Python.

Introdução

OBS. Reforça-se que:

1. Podem existir zero ou mais cláusulas elif e a cláusula final else é opcional.
2. Caso se queira avaliar um caso particular, mas não executar nenhum código devemos usar `pass`, que faz com que o programa nada execute.
3. O valor `0` (zero) equivale a falso para o python, enquanto que qualquer valor não nulo equivale a um verdadeiro.

Observe o código abaixo:

```
A=0
B=1
if A:
    pass #Este bloco de código não tem como ser executado
else:
    print("0 é equivalente a falso no python.")
if B:
    print("Inteiro não nulo é equivalente a verdade no python.")
else:
    pass
```

A saída do programa é:

```
0 é equivalente a falso no python.
Inteiro não nulo é equivalente a verdade no python.
```

Introdução

OBS. Reforça-se que:

1. Podem existir zero ou mais cláusulas elif e a cláusula final else é opcional.
2. Caso se queira avaliar um caso particular, mas não executar nenhum código devemos usar `pass`, que faz com que o programa nada execute.
3. O valor `0` (zero) equivale a falso para o python, enquanto que qualquer valor não nulo equivale a um verdadeiro.

Observe o código abaixo:

```
A=0
B=1
if A:
    pass #Este bloco de código não tem como ser executado
else:
    print("0 é equivalente a falso no python.")
if B:
    print("Inteiro não nulo é equivalente a verdade no python.")
else:
    pass
```

A saída do programa é:

```
0 é equivalente a falso no python.
Inteiro não nulo é equivalente a verdade no python.
```

Introdução

OBS. Reforça-se que:

1. Podem existir zero ou mais cláusulas elif e a cláusula final else é opcional.
2. Caso se queira avaliar um caso particular, mas não executar nenhum código devemos usar `pass`, que faz com que o programa nada execute.
3. O valor `0` (zero) equivale a falso para o python, enquanto que qualquer valor não nulo equivale a um verdadeiro.

Observe o código abaixo:

```
A=0
B=1
if A:
    pass #Este bloco de código não tem como ser executado
else:
    print("0 é equivalente a falso no python.")
if B:
    print("Inteiro não nulo é equivalente a verdade no python.")
else:
    pass
```

A saída do programa é:

```
0 é equivalente a falso no python.
Inteiro não nulo é equivalente a verdade no python.
```

Introdução

OBS. Reforça-se que:

1. Podem existir zero ou mais cláusulas elif e a cláusula final else é opcional.
2. Caso se queira avaliar um caso particular, mas não executar nenhum código devemos usar `pass`, que faz com que o programa nada execute.
3. O valor `0` (zero) equivale a falso para o python, enquanto que qualquer valor não nulo equivale a um verdadeiro.

Observe o código abaixo:

```
A=0
B=1
if A:
    pass #Este bloco de código não tem como ser executado
else:
    print("0 é equivalente a falso no python.")
if B:
    print("Inteiro não nulo é equivalente a verdade no python.")
else:
    pass
```

A saída do programa é:

```
0 é equivalente a falso no python.
Inteiro não nulo é equivalente a verdade no python.
```

Introdução

OBS. Reforça-se que:

1. Podem existir zero ou mais cláusulas elif e a cláusula final else é opcional.
2. Caso se queira avaliar um caso particular, mas não executar nenhum código devemos usar `pass`, que faz com que o programa nada execute.
3. O valor `0` (zero) equivale a falso para o python, enquanto que qualquer valor não nulo equivale a um verdadeiro.

Observe o código abaixo:

```
A=0
B=1
if A:
    pass #Este bloco de código não tem como ser executado
else:
    print("0 é equivalente a falso no python.")
if B:
    print("Inteiro não nulo é equivalente a verdade no python.")
else:
    pass
```

A saída do programa é:

```
0 é equivalente a falso no python.
Inteiro não nulo é equivalente a verdade no python.
```

Introdução

OBS. Reforça-se que:

1. Podem existir zero ou mais cláusulas elif e a cláusula final else é opcional.
2. Caso se queira avaliar um caso particular, mas não executar nenhum código devemos usar `pass`, que faz com que o programa nada execute.
3. O valor `0` (zero) equivale a falso para o python, enquanto que qualquer valor não nulo equivale a um verdadeiro.

Observe o código abaixo:

```
A=0
B=1
if A:
    pass #Este bloco de código não tem como ser executado
else:
    print("0 é equivalente a falso no python.")
if B:
    print("Inteiro não nulo é equivalente a verdade no python.")
else:
    pass
```

A saída do programa é:

```
0 é equivalente a falso no python.
Inteiro não nulo é equivalente a verdade no python.
```

Sumário

BeeCrowd Programa 1006

beecrowd | 1006

Média 2

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Leia 3 valores, no caso, variáveis A, B e C, que são as três notas de um aluno. A seguir, calcule a média do aluno, sabendo que a nota A tem peso 2, a nota B tem peso 3 e a nota C tem peso 5. Considere que cada nota pode ir de 0 até 10.0, sempre com uma casa decimal.

Entrada

O arquivo de entrada contém 3 valores com uma casa decimal, de dupla precisão (double).

Saída

Imprima a mensagem "MEDIA" e a média do aluno conforme exemplo abaixo, com 1 dígito após o ponto decimal e com um espaço em branco antes e depois da igualdade. Assim como todos os problemas, não esqueça de imprimir o fim de linha após o resultado, caso contrário, você receberá "Presentation Error".

Exemplos de Entrada	Exemplos de Saída
5.0 6.0 7.0	MEDIA = 6.3
5.0 10.0 10.0	MEDIA = 9.0

Sumário

Aplicação

Resolva o problema 1013 da `beecrowd` utilizando uma estrutura condicional e com a sugestão dada.

URI Online Judge | 1013

O Maior

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Faça um programa que leia três valores e apresente o maior dos três valores lidos seguido da mensagem "eh o maior". Utilize a fórmula:

$$MaiorAB = \frac{(a+b+abs(a-b))}{2}$$

Obs.: a fórmula apenas calcula o maior entre os dois primeiros (a e b). Um segundo passo, portanto é necessário para chegar no resultado esperado.

Entrada

O arquivo de entrada contém três valores inteiros.

Saída

Imprima o maior dos três valores seguido por um espaço e a mensagem "eh o maior".

Exemplos de Entrada	Exemplos de Saída
7 14 106	106 eh o maior
217 14 6	217 eh o maior

Sumário

Aplicação

Resolva o problema 1036 da `beecrowd`.

beecrowd | 1036

Fórmula de Bhaskara

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Leia 3 valores de ponto flutuante e efetue o cálculo das raízes da equação de Bhaskara. Se não for possível calcular as raízes, mostre a mensagem correspondente *"Impossivel calcular"*, caso haja uma divisão por 0 ou raiz de numero negativo.

Entrada

Leia três valores de ponto flutuante (double) A, B e C.

Saída

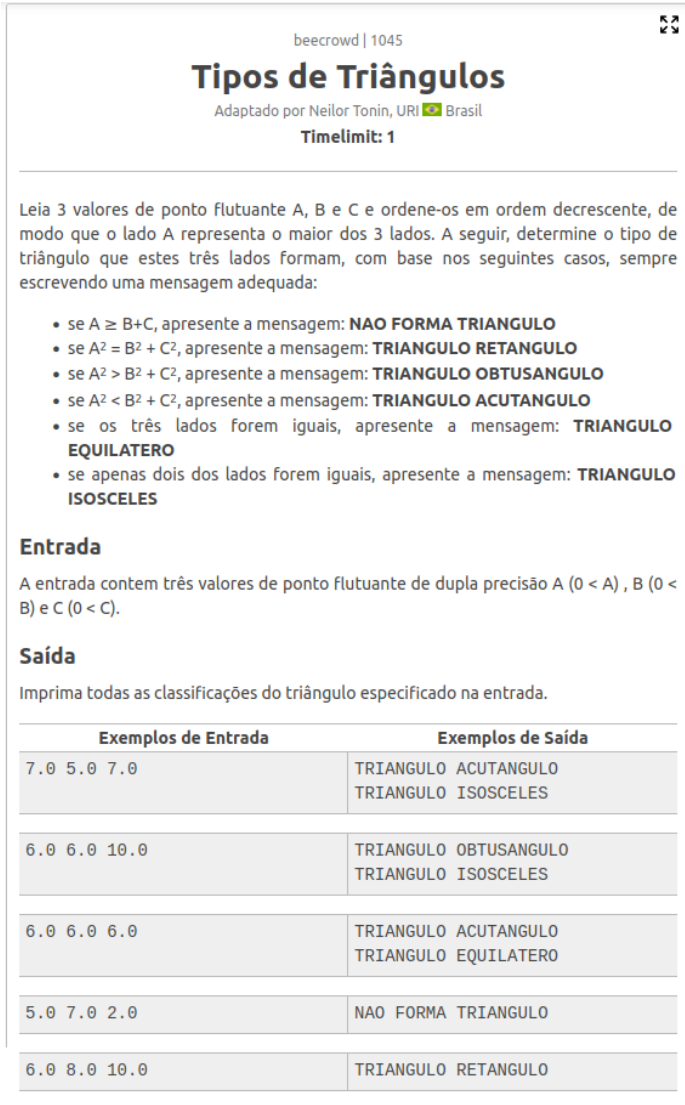
Se não houver possibilidade de calcular as raízes, apresente a mensagem "Impossivel calcular". Caso contrário, imprima o resultado das raízes com 5 dígitos após o ponto, com uma mensagem correspondente conforme exemplo abaixo. Imprima sempre o final de linha após cada mensagem.

Exemplos de Entrada	Exemplos de Saída
10.0 20.1 5.1	R1 = -0.29788 R2 = -1.71212
0.0 20.0 5.0	Impossivel calcular
10.3 203.0 5.0	R1 = -0.02466 R2 = -19.68408
10.0 3.0 5.0	Impossivel calcular

Sumário

Aplicação

Resolva o problema 1045 da `beecrowd`.



beecrowd | 1045

Tipos de Triângulos

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

Leia 3 valores de ponto flutuante A, B e C e ordene-os em ordem decrescente, de modo que o lado A representa o maior dos 3 lados. A seguir, determine o tipo de triângulo que estes três lados formam, com base nos seguintes casos, sempre escrevendo uma mensagem adequada:

- se $A \geq B+C$, apresente a mensagem: **NAO FORMA TRIANGULO**
- se $A^2 = B^2 + C^2$, apresente a mensagem: **TRIANGULO RETANGULO**
- se $A^2 > B^2 + C^2$, apresente a mensagem: **TRIANGULO OBTUSANGULO**
- se $A^2 < B^2 + C^2$, apresente a mensagem: **TRIANGULO ACUTANGULO**
- se os três lados forem iguais, apresente a mensagem: **TRIANGULO EQUILATERO**
- se apenas dois dos lados forem iguais, apresente a mensagem: **TRIANGULO ISOSCELES**

Entrada

A entrada contem três valores de ponto flutuante de dupla precisão A (0 < A) , B (0 < B) e C (0 < C).

Saída

Imprima todas as classificações do triângulo especificado na entrada.

Exemplos de Entrada	Exemplos de Saída
7.0 5.0 7.0	TRIANGULO ACUTANGULO TRIANGULO ISOSCELES
6.0 6.0 10.0	TRIANGULO OBTUSANGULO TRIANGULO ISOSCELES
6.0 6.0 6.0	TRIANGULO ACUTANGULO TRIANGULO EQUILATERO
5.0 7.0 2.0	NAO FORMA TRIANGULO
6.0 8.0 10.0	TRIANGULO RETANGULO